НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК *519.68;681.513.7*

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*18* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *122 Комп'ютерні науки та інформаційні технології*
(код та назва спеціальності)

спеціалізацією *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Інформаційна технологія автоматичної класифікації неструктурованих текстових масивів інформації*

Виконав: студент *VI* курсу групи ІС-61м
(шифр групи)

Катющенко Дар’я Олександрівна _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *старший викладач Олійник Ю.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

РЕФЕРАТ

Магістерська дисертація: 99 с., 19 рис., 22 табл., 1 додаток, 63 джерела.

Актуальність. Потреба в автоматичній обробці текстових документів зараз є надзвичайно високою, і постійно зростає. Це обумовлено щоденним збільшенням текстової інформації на просторах всесвітньої мережі інтернет. За даними на березень 2016 року в Інтернеті знаходиться близько 4,66 млрд сторінок, при чому ця цифра включає лише сторінки, які індексовані в найбільш розповсюджених пошукових системах. Тож, без комп'ютерної обробки виконати аналіз такого об'єму інформації за прийнятний час не можливо.

Одною із задач інтелектуального аналізу текстів є їх класифікація на задані категорії, яка потребує вирішення в різних сферах людської діяльності. Так, для забезпечення інформаційної та суспільної безпеки, важливе значення має аналіз даних соціальних мереж, блогів тощо, з метою виявлення даних пов'язаних з тероризмом, наркоторгівлею і т.д. Також в комерційній та суспільній діяльності часто постає потреба обробки відгуків та коментарів, з метою виявлення їх емоційного забарвлення (негативного або позитивного), їх розподіл на подальше опрацювання між різними підрозділами і т.д. В першому та в другому прикладах постає задача класифікації текстової інформації між категоріями в умовах обмеженості за часом та ресурсами обчислювальних пристроїв. Тому задача автоматичної та якісної класифікації даних за прийнятний час, без попереднього структурування інформації, оскільки структурування потребує додаткових ресурсів, часу та не завжди може пройти без втрати важливої інформації, є задачею, яка варта уваги та досліджень.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Методи та

технології високопродуктивних обчислень та обробки надвеликих масивів даних». Державний реєстраційний номер 0117U000924.

Метою дослідження є покращення якості моніторингу медіа активності шляхом розробки алгоритму автоматичної класифікації текстової інформації, що дозволяє підвищити точність та повноту аналізу даних.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– проаналізувати алгоритми та методи машинного навчання для вирішення задачі автоматичної класифікації текстів;

– обрати модель представлення текстової інформації в класифікаторі;

– розробити алгоритм попередньої обробки текстів відповідно до обраної моделі представлення текстової інформації;

– розробити модифікований метод класифікації текстової інформації;

– виконати програмну реалізацію розробленого алгоритму автоматичної класифікації текстової інформації;

– підготувати данні для оцінки якості класифікації;

– провести дослідження ефективності розробленої інформаційної технології.

Об’єктом дослідження є процес класифікації неструктурованих текстових масивів інформації.

Предметом дослідження є технології та методи інтелектуального аналізу текстової інформації.

Методами дослідження є методи машинного навчання, які базуються на методах text mining.

Наукова новизна отриманих результатів. Розроблено модифікований метод індексації на основі статистичного алгоритму Вітербі з підключенням бази граматичних правил зняття морфологічної омонімії.

Публікації. Результати досліджень опубліковані в журналі «Науковий огляд» [1],опубліковані в тезах науково практичної конференції математичне та

імітаційне моделювання систем. МОДС "2017" [2], опубліковано в тезах наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка» – ІОТ-2018 [3].

МАШИННЕ НАВЧАННЯ, TEXT MINING, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ТЕКСТОВИЙ КЛАСИФІКАТОР, АНАЛІЗ КОНТЕНТУ, АЛГОРИТМИ КЛАСИФІКАЦІЇ, ІНДЕКСАЦІЯ ТЕКСТІВ

ABSTRACT

Master dissertation: 99 pp., 19 fig., 22 tab., 1 app., 63 sources.

Topicality. The need for automatic processing of text documents is now extremely high and is steadily increasing. This is due to the daily increase of textual information on the expanses of the World Wide Web. According to March 2016, there are about 4.66 billion pages on the Internet, and this number only includes pages that are indexed in the most widely used search engines. Therefore, without computer processing, it is not possible to analyze such information volumes in a reasonable time.

One of the tasks of intellectual analysis of texts is their classification on the given categories, which needs to be solved in different spheres of human activity. So, for the purpose of securing information and public security, the analysis of social network data, and everyday blogs is important for the purpose of identifying data related to terrorism, drug trafficking, etc. Also, in commercial and public activities, there often is a need to process reviews and comments, in order to identify their emotional color (negative or positive), their distribution on further processing between different Subdivisions, etc. In the first and second examples, the task of classifying textual information among categories in terms of time constraints and resources of computing devices arises. Therefore, the task of automatic and qualitative classification of data at an acceptable time, without prior structuring of information, since structuring requires additional resources, time and can not always go through without loss of important information, is a task that is worth attention and research.

Relationship of work with scientific programs, plans, themes. The research was carried out at the Department of Computer-Aided Management And Data Processing Systems of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» within the theme «Methods and Technologies for High-Performance Computing and Processing of Large-Size Data Arrays.» State registration number 0117U000924.

The aim of the research is to improve the quality of monitoring of media activity by developing an algorithm for automatic analysis of text information, which increases the accuracy and completeness of the classification.

To achieve this goal, the following tasks must be performed:

– analyze algorithms and methods of machine learning for solving the problem of automatic classification of texts;

– choose a model for presenting textual information in the classifier;

– develop an algorithm for pre-processing texts in accordance with the selected model of presentation of textual information;

– develop a modified method for classifying textual information;

– implement the program realization of the developed algorithm of automatic classification of textual information;

– prepare data for the assessment of the quality of the classification;

– to conduct research on the effectiveness of the developed information technology.

The object of research is the process of classifying unstructured text arrays of information.

The subject of research there are technologies and methods of intellectual analysis of textual information.

Research methods are methods of machine learning, which based on text mining methods.

Scientific novelty of the obtained results. The modified indexing method on the basis of the statistical algorithm Viterbi with the connection of the grammatical rules of removal of morphological homonymy has been developed.

Publications. The results of the research are published in the journal "Scientific Review" [1] published in the theses of the scientific and practical conference mathematical and simulation modeling systems. MODIS "2017" [2], published in the

theses of the scientific conference of students, graduate students and postgraduates "Informatics and Computing" - IOT-2018 [3].

MACHINE LEARNING, DATA MINING, TEXT MINING, CLASSIFICATION OF TEXTUAL INFORMATION, CONTENT ANALYSIS, CLASSIFICATION ALGORITHMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Терм (ознака)	–	значиме слово або словосполучення документу.
VSM	–	англ. Vector Space Model – модель представлення тексту у вигляді вектору.
BEP	–	англ.breakevenpoint – точка беззбитковості.
NER	–	англ. Named Entity Recognition – розпізнання іменованих сутностей.
IDC	–	англ. International data corporation.
TP	–	істинно-позитивне рішення.
TN	–	істинно-негативне рішення.
FP	–	хибно-негативне рішення.
FN	–	хибно-негативне рішення.
MaxEnt	–	англ. Maximum entropy – метод максимума ентропії.
CBOW	–	англ. Continuous Bag of Words.
IG	–	англ. Information gains – інформаційний приріст.
PMI	–	англ. Pointwise Mutual Information – покрапкова взаємна інформація

ВСТУП

Комп’ютеризація усіх сфер життя обумовлює збільшення об’ємів інформації. Загальний об’єм даних в світі в 2011 р. складав більше 1,8 зеттабайт (1,8 трлн Гб) та за дослідженнями IDC ця цифра подвоюється кожні 2 роки. Збереження такого об’єму інформації потребує немалих ресурсів та зусиль. Поява централізованих сховищ даних та накопичення в них інформації у вигляді як структурованих таблиць так і слабоструктурованих та неструктурованих текстів призвели до росту уваги до методів аналізу даних. Аналіз даних дозволяє прийняти вірне рішення чи спрогнозувати подальший розвиток подій. За прогнозами IDC в 2020 році з усіх об’ємів даних лише 35% будуть корисними. Тому все більше виникає потреба в обробці потоків даних в реальному часі, в тому рахунку й текстових.

Однією із задач аналізу даних є їх класифікація за заданими категоріями, тобто визначення належності тексту до деякої категорії в умовах постійного росту об’ємів інформації, яка обробляється.

Класифікація застосовується для рішення наступних задач:

- розпізнання спаму;
- моніторинг новин;
- автоматичне анотування;
- фільтрація документів;
- пошук текстових документів;
- навігація по великим інформаційним ресурсам;
- підбір реклами;
- складення каталогів;
- індексування даних в пошукових системах та багато інших.

Також автоматична класифікація застосовується для забезпечення інформаційної та суспільної безпеки. Важливе значення має аналіз даних

соціальних мереж та блогів, з метою виявлення даних пов’язаних з тероризмом, наркоторгівлею, тощо. Автоматична класифікація має значення й в комерційній та суспільній діяльності, наприклад, для обробки відгуків або коментарів, з метою виявлення їх емоційного забарвлення (негативного або позитивного), чи їх розподіл на подальше опрацювання між різними підрозділами. В усіх перелічених вище задачах виникає потреба в класифікації текстової інформації між різними категоріями в умовах обмеженості за часом та ресурсами обчислювальних пристроїв. Тому задача автоматичної та якісної класифікації даних за прийнятний час є задачею, яка варта уваги та досліджень.

З метою покращення моніторингу медіа активності, через підвищення точності та повноти класифікації, в даній роботі було виконано ряд наступних завдань:

– аналіз алгоритмів та методів машинного навчання для вирішення задачі автоматичної класифікації текстів;

– аналіз проблем попередньої обробки текстів, які впливають на якість класифікації;

– розробка модифікованого алгоритму рішення однієї з таких проблем для класифікації англомовних текстів, а саме – зняття морфологічної омонімії;

– проектування та реалізація інформаційної технології для класифікації текстів та виконання її порівняльного аналізу з існуючими популярними методами класифікації.

Морфологічна омонімія — це явище, яке полягає в збіганні однієї або декількох граматичних форм слів, які належать різним граматичним класам. Англійська мова має чітку структуру речення, тому на відміну від більш гнучких мов, в яких до граматичних класів належать частина мови, відмінки, рід тощо; в англійській мові присутня лише частиномовна омонімія.

Існує два підходи для зняття морфологічної омонімії. Один з них — детермінований, заснований на граматичних правилах та словниках, а інший — статистичний, заснований на статистиці попередніх результатів.

В даній роботі розглянуто обидва підходи, їх переваги та недоліки та запропоновано гібридний алгоритм рішення зняття омонімії в англомовних текстах.

1 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ НЕСТРУКТУРОВАНИХ ТЕКСТОВИХ МАСИВІВ ІНФОРМАЦІЇ

1.1 Етапи автоматичної класифікації текстової інформації

Розв'язок задачі класифікації неструктурованих текстових масивів інформації складається з наступних 4-х послідовних етапів (рисунок 1.1):

– попередня обробка та індексація документів;

– зменшення розмірності множини ознак;

– побудова та навчання класифікатора за допомогою методів машинного навчання;

– оцінка якості класифікації.

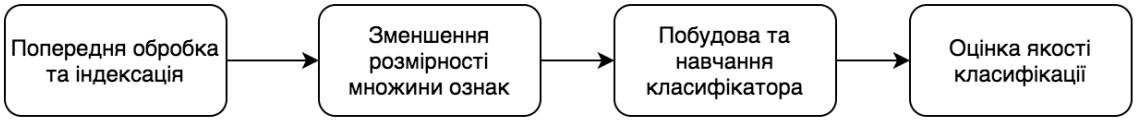

Рисунок 1.1 – Етапи класифікації

Етап попередньої обробки призначений для перетворення документів на природній мові в зручну для машинної обробки форму, тобто формування ознак документа, в якості яких виступають всі його значимі слова [4]. Основними складовими цього етапу є: токенізвція, лінералізація та нормалізація.

Токенізація. На етапі токенізації виконується розбиття тексту на більш дрібні об'єкти, наприклад, речення, фрази або слова.

Лінералізація. Етап лінералізація полягає в видалення зайвих символів, стоп слів, вилучення термів та термових словосполучень або n-грам. N-грама – послідовність n символів, які йдуть один за одним, виду $a_1,\ldots,a_{n-1},a_n$. Символи a_i належать деякому алфавіту розмірності M.

Нормалізація. Етап нормалізація текстового документу полягає в:

– приведенні слів документу до одного регістру;

– видалення відмін, числа, часів тощо, в залежності від мови тексту;

– визначення частини мови слів за рахунок врахування контексту та багаточисельних правил узгодження слів.

Нормалізація може включати в себе наступні компоненти[5]:

– стематизацію – процес відшукання основи слова, при чому псевдооснова, яка буде отримана, не має обов'язково збігатися з граматичною основою слова;

– лематизацію – процес, що приводить словоформу до леми – її нормальної(словникової) форми ;

– коментування (присвоювання грамем) – присвоювання граматичних характеристик, які відносять словоформу до певного морфологічного класу.

По завершенню етапу попередньої обробки документ представляється в одному з наступних виглядів, що розглянуті нижче[6].

VSM. Це модель, в якій текстовий документ буде представлений у вигляді вектору, кожна координата якого відповідає вазі терма (зазичай частоті зустрічання цього терма в текстовому документі). По завершенню обробки вибірки формується прямокутна матриця розмірності $n \times m$, де n – кількість документів, m – кількість термів в колекції. Таке представлення тексту не потребує використання додаткових попередніх аналізів, таких як визначення частини мови, присвоєння грамем тощо.

Поліграмна модель. В поліграмній моделі зі ступенем n та основою M текст представляється вектором $\{f_i\} i = \overline{1, M}$, де f_i – частота зустрічання i –ої n-грами в тексті. Вважається, що частота з'явлення n-грами в тексті несе важливу інформацію про властивості документа. Крім того максимальна кількість n-грам постійної довжини для конкретної мови є фіксованою та не залежить від об'єму навчальної вибірки. Однак, як показують практичні дослідження в реальних

текстах реалізується не більше 25-30 відсотків від загальної їх допустимої кількості. Перевагами поліграмної моделі є те, що немає необхідності додаткової лінгвістичної обробки тексту, фіксована розмірність векторів та простота отримання векторного опису тексту. Однак відображення змісту тексту векторами $\{f_i\}$ не завжди є адекватним (такий підхід погано відображає зміст невеликих текстів), тому модель більше підходить для визначення мови ніж для класифікації по тематиці [7].

Лінгвістична розмітка. Модель, яка представляє текстові документи з синтаксичними характеристиками, тобто задає інформацію про лінгвістичні одиниці безпосередньо в тексті в формі розмітки на таких мовах, як SGML або XML. Для приведення тексту до такого представлення використовують синтаксичний або морфологічний аналіз (присвоєння грамем).

Дерева залежностей. Граматика залежностей передбачає, що речення тексту можна структурувати у вигляді дерев залежностей, в яких слова зв'язані орієнтованими дугами, які визначають синтаксичний зв'язок між головним та підлеглим словами.

Дерева складових. Це модель формалізації тексту, в якій слова в реченні природної мови групуються в складові на основі лінгвістичних спостережень того, що ланцюжки слів в реченні можуть функціонувати як одне ціле та підкорятися одним граматичним правилам. Складові можна перенести в середину або в кінець речення цілком, але частково без втрати змісту їх перенести не можливо.

Детальний розбір побудови дерева залежностей наведений в [5]. Відмітимо лише, що, по-перше, дерева залежностей мають менший розмір в порівнянні з деревами складових, оскільки зазвичай мають меншу кількість вершин. По-друге, на відміну від дерев складових, дерева залежностей відображають не реальний порядок слів в реченні, а лише ієрархію зв'язків. По-третє, в деревах залежностей дієслово є основою всієї структури речення, в той час як дерева складових

будуються на правилах бінарного розподілу. Тож дерева залежностей краще відображають специфіку мов, в яких немає чіткого порядку слів у реченнях.

Етап зменшення розмірності множини ознак є досить важливим, оскільки обчислювальна складність методів класифікації напряму залежить від розміру множини термів, тому для ефективної роботи класифікатора часто вдаються до скорочення числа використовуваних ознак. Зменшення розмірності множини ознак - це процес надання ваги словам, в залежності від їх важливості для класифікації тексту, та видалення із множини ознак мало важливих термів. Важливість слів визначається їх здатністю характеризувати відмінність між категоріями в корпусі документів.

За рахунок зменшення розмірності множини термінів можна знизити ефект перенавчання – явище за якого класифікатор орієнтується на випадкові або помилкові характеристики навчаючих даних, а не на насправді важливі.[9]

Для етапу побудови та навчання класифікатора існує велика кількість методів вибір яких залежить від поставленої задачі та способу формалізації документу, обраного на першому етапі.

Загалом класифікатори можна поділити на лінійні та нелінійні[10]. Лінійні класифікатори побудовані на простій лінійній комбінації функцій. Такі класифікатори поділяють простір ознак на регіони за допомогою лінійних гіперплощин.

Нелінійні – більш складні класифікатори, але не завжди кращі на лінійні, через зміщення дисперсії. Такі моделі мають багато параметрів, які підходять для обмеженої кількості навчальних даних, але з більшою ймовірністю припустяться помилки на невеликих та шумних наборах даних.

На етапі оцінки якості класифікації використовують тестову вибірку даних, тобто набір текстів, для яких відомі обрані експертами класи, таких щоб навчальна та тестова вибірки не пересікалися. Якщо спочатку знайти оптимальні ознаки по всій вибірці, а потім оцінювати якість алгоритму, то відібрані ознаки

вже оптимізують якість, тому оцінка буде занадто оптимістичною. Щоб оцінка якості класифікатора була об'єктивною, необхідно правильно вибрати співвідношення обсягів навчальної та тестової вибірок. Якщо взяти дуже маленьку навчальну вибірку, оцінка якості буде дуже песимістичною. Якщо тестова вибірка буде маленька, оцінка виявиться неточною. Як правило, навчальну та тестову вибірки беруть виходячи зі співвідношення 70/30[11].

1.2 Огляд методів індексації документів

Найбільш популярними методами індексації текстів зараз є: Bag-of-words[14],Word2Vec [14,15], n-grams[16].

Bag-of-words– модель, яка представляє тексти як набір слів без урахування їх порядку та зв’язків, тому при використанні цього методу семантичну близькість текстів оцінюють за кількістю співпадаючих слів. Таким чином тексти, які містять малу кількість спільних слів, вважаються семантично далекими один від одного. Отже Bag-of-word не враховує семантичні зв’язки слів, а тексти, як набори слів, проектується в простір високої розмірності й високої розрідженості, що обумовлено об’ємом словника, який використовується.

Існують підходи для подолання цього недоліку, які можна розділити на 3 групи [16]:

- підходи на основі концептів (Bag-of-concept);
- підходи на основі концептних векторів;
- підходи на основі латентно-семантичної моделі.

Підходи на основі концептів[17–24] представляють тексти за допомогою концептів - категорій, які об’єднують семантично зв’язані слова. Під концептами розуміються не лише ряди синонімів, але й семантичні та тематичні кластери.

Концепти та тексти індексуються за допомогою слів, тобто кожному концепту відповідає вектор $\overline{C} = (c_1, c_2, ..., c_n)$, де n – об'єм словника, а $c_i (i = \overline{1,n})$ – вага слів в концепті.

Аналогічно для текстів співвідноситься вектор $\overline{D} = (w_1, w_2, ..., w_n)$, де $w_i (i = \overline{1,n})$ – вага слів в тексті. Близькість між концептом та текстом можна визначити за допомогою наступних трьох формул:

– на основі косинусової міри:

$$Sim(C, D) = \frac{\sum_{i=1}^{n} c_i w_i}{\sqrt{\sum_{i=1}^{n} w_i^2} \sqrt{\sum_{i=1}^{n} c_i^2}} \tag{1.7}$$

– на основі коефіцієнта кореляції Спірмена (на скільки співпадає порядок слів в концепті та тексті):

$$Corr(C, D) = 1 - \frac{6\sum_{i=1}^{T} (rank(w_i, C) - rank(w_i, D))^2}{T(T^2 - 1)} \tag{1.8}$$

де T - це кількість слів, які одночасно входять і в текст і в концепт;

rank - функція, яка повертає ранг (позицію) заданого слова в тексті або концепті.

– на основі контекстної та рангової близькості:

$$\mathrm{Re}l(C, D) = (1 + Sim(C, D)) * (1 + Corr(C, D)) \tag{1.9}$$

Після визначення близькості між концептом та текстом, визначається набір з k найбільш близьких тексту концептів.

Підходи на основі концептних векторів [25,26] формують так звані контекстні вектори слів, які показують залежність даного слова від всіх інших слів простору, який використовується. В той час як, моделі концептів враховує не всі, а лише обрані залежності між словами.

Контекстні вектори n вимірного простору, які записані разом, утворюють квадратну матрицю "терміни-на-терміни" розмірності $n \times n$. Таку матрицю також

часто називають матрицею попарних зв'язків чи матрицею залежностей. Кожен її елемент $(i, j)(i, j = \overline{1,n})$ – чисельне представлення залежності між *i*–м та *j*–м словами (термінами).

Підходи на основі латентної семантичної моделі створюють представлення текстів за допомогою латентних тематик. Формування таких представлень здійснюються шляхом факторизації матриці "документ $\times$ терміни".

Найбільш популярним методом серед даного класу є латентний семантичний аналіз (LSA). В якості вхідної інформації LSA використовує матрицю терми $\times$ документи з набором даних, який використовується для навчання системи. Елементи цієї матриці як правило містять ваги термів, які обчислюються на основі частоти використання кожного терма в кожному документі та наявності цього терма в усіх документах. LSA в якості факторизації використовує сингулярне розкладення [0]. Сингулярне розкладення матриці дозволяє замінити вихідну матрицю розмірності $m \times n$ ортогональними матрицями розмірностей $m \times r$ та $r \times n$ відповідно та діагональною матрицею розмірності $r \times r$, де $r = \min(m, n)$:

$$A_{(m \times n)} = U_{(m \times r)} * S_{(r \times r)} * V^T_{(r \times n)}, \tag{1.10}$$

де $U_{(m \times r)}$ та $V_{(r \times n)}$ – ортогональні, а $S_{(r \times r)}$ – діагональна.

Такий розподіл має наступну найголовнішу особливість: якщо в матриці $S_{(r \times r)}$ залишити тільки *k* найбільших сингулярних значень, а в матрицях $U_{(m \times r)}$ та $V_{(r \times n)}$ тільки відповідні цим значенням стовпці, то добуток матриць S, U та V буде найкращим наближенням матриці *A* до матриці $\hat{A}$ рангу *k*:

$$\hat{A} \approx A = USV^T \tag{1.11}$$

На цьому заснована основна ідея латентно-семантичного аналізу, в якому в якості *A* використовується матриця терми $\times$ документи. Тобто $\hat{A}$, яка містити тільки *k* перших лінійной-незалежних компонент *A*, відображає основну

структуру залежностей, які містяться у висхідній матриці. структура залежностей визначається ваговими функціями термів. [28]

Тобто така заміна дозволяє позбутися більшої частини малозначимої та випадкової інформації, натомість виявляються приховані тематичні тренди. Загалом ці приховані тренди позначають ті ж самі тематичні концепти, тільки вилучені іншим способом.

Таким чином кожний терм та документ представляються за допомогою векторів в загальному просторі *k* (просторі гіпотез). Близькість між будь-якою комбінацією термів або документів легко обчислюється за допомогою скалярного добутку векторів.

Окрім LSA, використовують метод невід'ємної матричної факторизації [29], який заснований на тому ж принципі вилучення прихованих тематик.

Word2Vec. Word2vec приймає великий текстовий корпус в якості вхідних даних і зіставляє кожному слову вектор, видаючи координати слів на виході. Спочатку він створює словник, «навчаючись» на вхідних текстових даних, а потім обчислює векторне подання слів. Векторне подання ґрунтується на контекстній близькості: слова, що зустрічаються в тексті поруч з однаковими словами (а отже, мають схожий зміст), у векторному поданні матимуть близькі координати векторів-слів. Отримані вектори-слова можуть бути використані для обробки природної мови та машинного навчання.

Отримувані на виході координатні уявлення векторів-слів дозволяють обчислювати «семантичну відстань» між словами. І, саме ґрунтуючись на контекстній близькості цих слів, технологія word2vec здійснює свої передбачення.

У word2vec існують два основних алгоритми навчання: CBOW [30] і Skip-gram[31]. CBOW - модельна архітектура, яка передбачає поточне слово, виходячи з навколишнього його контексту. Тобто використовує слова, які оточують поточне (речення без поточного слова), для того щоб передбачити поточне слово.

На вхід подаються вектори: $W(t-2), W(t-1), W(t+1), W(t+2)$.

На виході отримують вектор $W(t)$.

Архітектура типу Skip-gram діє навпаки: вона використовує поточне слово, щоб передбачати навколишні його слова.

Обидві ці моделі використовують нейрону мережу для відображення слів, які ближче до заданого слова за значенням.

N-grams. Також поширеним підходом індексації документів, є підхід заснованій на обліку n-грам[32]. З семантичної точки зору N-грами зустрічаються у вигляді ряду слів, стійкі словосполучення називають колокацією.

N-грами використовується в основному для передбачення на основі імовірнісних моделей. N-грамна модель розраховує ймовірність останнього слова N-грами, якщо відомі всі попередні. При використанні цього підходу для моделювання мови передбачається, що поява кожного слова залежить тільки від попередніх слів.

Для швидкого вилучення n-грам з тексту використовують наступні методи:

– алгоритм Лемпеля - Зива - Велча (LZW) - це універсальний алгоритм стиснення без втрат. Його модифікації (LZMW та LZAP) [33];

– суфіксний масив [34];

– суфіксне дерево [34,35];

– інвертований індекс [36].

Як відзначено в [5] серед проблем які потребують вирішення на етапі попередньої обробки текстів та індексації документів є проблема NER в цілому, а зокрема наступні проблеми етапу попередньої обробки текстів:

– морфологічна омонімія — збігання однієї або декількох граматичних форм слів в написанні або мовленні, які належать різним частинам мови, наприклад, *рукав* (елемент одягу) — *рукав* (річки), *close* – закривати та *close* – близький;

– синтаксична омонімія — неоднозначність, яка виникає через неясність синтаксичних зв’язків між словами у реченні, наприклад, викладач запропонував прийти на залік у вівторок (Запропонував в вівторок чи прийти в вівторок?);

– лексична омонімія та полісемія — збігання однієї або декількох граматичних форм слів в письмі та вимові, слова належать до однієї частини мови, але мають різні значення, наприклад, *lead* – свинець та *lead* – лідерство (в такому випадку слова сильно залежать від контексту);

– синтаксична синонімія — явище за якого синтаксичні конструкції мають близькі значення та здатні в деяких контекстах замінити один одного;

– вирішення кореферентності — визначення відносин між компонентами речення, в якому слова посилаються на одні й ті самі об’єкти, наприклад, if *Tom and Jim* won’t make their homework, *their parents* will be upset, and don’t let *them* play games till Friday.

З метою позбавлення від морфологічної омонімії застосовують детермінований та статистичний підходи. Детермінований заснований на локальному та глобальному синтаксичному розборі, синтаксичних словниках та на правилах узгодження слів. Такий підхід потребує великих затрат часу та ресурсів. Статистичний підхід використовує статистику спільної зустрічності слів в великих корпусах, омонімія в яких знята заздалегідь. Він виконується на етапі морфологічного аналізу. Найпоширенішими статистичними методами зняття морфологічної омонімії є HMM (прихована марківська модель), MEMM (Марковські моделі із максимальною ентропією) та алгоритм Вітербі [37].

В якості алгоритму стематизації текстів часто використовують стемінг Портера. Цей алгоритм не використовує баз основ слів, а працює, послідовно застосовуючи ряд правил відсікання закінчень і суфіксів у словах для отримання основи слова.

1.3 Огляд методів визначення ваг ознак документів

До найбільш розповсюджених методів визначення ваг ознак документа відносяться: TF-IDF [38],TF-SLF[37] критерій χ^2(CHI)[39], по-крапкова взаємна інформація (PMI)[40], умовні випадкові поля (CRF)[41], інформаційний приріст (IG)[40].

TF-IDF. Цей метод заснований на двох правилах:

– term frequency — чим частіше терм t_k зустрічається в документі d_j, тим він більш значимий;

– inverse document frequency — чим в більшій кількості документів терм t_k зустрічається, тим він є менш відмінним.

Тож вага *i*-го терма в документі d_j визначається за наступною формулою:

$$w_{ij} = \frac{tf_{ij} * idf_i}{\sqrt{\sum_k (tf_{kj} * idf_k)^2}}, \tag{1.12}$$

де tf_{ij} — частота зустрічності *i*-го терма в документі, який розглядається;

$idf_i = \log\frac{N}{n}$ — логарифм співвідношення кількості документів в колекції до кількості документів, в яких зустрічається *i*-ий терм.

$$\sum_i w_{ij}^2 = 1. \tag{1.13}$$

Модифікацією методу TF-IDF можна вважати метод TF-SLF , який на відміну від TF-IDF, розглядає важливість терму в рамках окремо взятої категорії розглядає важливість терму в рамках кожної окремої категорії.

TF-SLF заснований на наступних положеннях:

– терм вважається вагомим в рамках даної категорії, якщо він зустрічається в більшості текстів цієї категорії;

– якщо терм є важливим для декількох категорій, то оцінка терму зменшується.

Нормалізована частота зустрічання терму *t* в категорії *c* обчислюється наступною формулою:

$$NDF_{tc} = \frac{df_{tc}}{N_c}, \tag{1.14}$$

де df_{tc}— кількість документів категорії *c*ьв яких хоча б раз зустрічається терм *t*;

N_c— кількість документів в категорії *c*.

NDF_{tc} є локальною оцінкою, глобальна оцінка терма розраховується наступним чином:

$$R_t = \sum_{c \in C} NDF_{tc}, \tag{1.15}$$

де *C*— множина категорій в корпусі документів.

Тоді загальна оцінка TF-SLF буде дорівнювати:

$$TFSLF_t = TF_t * SLF_t, \tag{1.16}$$

де $SLF_t = \log \frac{|C|}{R_t}$— логарифмована сума частот терму t.

Критерій χ^2. Даний метод дозволяє відповісти на питання чи впливає номінальний багаторівневий фактор на номінальний багаторівневий відгук. Загалом даний метод полягає в наступному: будується таблиця спорядженості, яка містить фактичні частоти, після чого знаходяться очікувані частоти, та виконують перевірку близькості фактичних та теоретичних частот. Якщо рівень вагомості менший за певний поріг, то фактор впливає на відгук. Заснований критерій χ^2 на перевірці незалежності двох подій. Нехай відбулося дві події A та B, вони будуть незалежними, якщо $P(A*B) = P(A)*P(B)$. В задачі формування множини ознак, подія A – це поява об’єкта з термом x_i, а подія B – це поява об'єкта з класом c_j. Тоді обчислити критерій χ^2 можна за наступною формулою:

$$\chi^2(D, x_i) = \sum_{x_j \in \{0,1\}} \sum_{c_j \in \{0,1\}} \frac{(\hat{V}_{x_i,c_j} - V_{x_i,c_j})^2}{V_{x_i,c_j}}, \tag{1.17}$$

де V – частота події $A \bigcap B$, а $\hat{V}$ – це очікувана частота у припущенні, що терм x_i і клас c_j незалежні. Критерій показує V та $\hat{V}$ будуть різними.

Метод інформаційного приросту (IG). Обчислює кількість бітів інформації отриманої для передбачення категорії через знання наявності або відсутності термів в документі.

Позначимо через $\{c_i\}_{i=1}^{m}$ набір категорій цільового простору. Тоді інформаційний приріст терма *t* визначається як:

$$IG(t) = -\sum_{i=1}^{m} P_r(c_i)\log P_r(c_i) + P_r(t)\sum_{i=1}^{m} P_r(c_i \mid t)\log P_r(c_i \mid t) + P_r(\bar{t})\sum_{i=1}^{m} P_r(c_i \mid \bar{t})\log P_r(c_i \mid \bar{t}) \tag{1.18}$$

В заданій навчальній вибірці для кожного окремого терму обчислюється *IG*, після чого видаляються ознаки, *IG* яких менше певного заданого порогу.

Методом покрапкової взаємної інформації (PMI). Семантичну близькість термінів можна також обчислити за допомогою PMI. Нехай задана таблиця, в який рядки відповідають термам *t* , а стовпці категоріям *c*. В цій таблиці *A* – кількість разів коли *t* та *c* зустрічаються разом, *B* – кількість разів, коли *t* зустрічається окремо від *c*, *C* – кількість разів, коли *c* зустрічається окремо від *t*, та *N* – загальна кількість документів вибірки. Тоді критерій спільної інформації між *t* та *c* визначається як:

$$I(t,c) = \log \frac{P_r(t \wedge c)}{P_r(t) \times P_r(c)}, \tag{1.19}$$

та обчислюється використовуючи співвідношення:

$$I(t,c) \approx \log \frac{A \times N}{(A+C) \times (A+B)}. \tag{1.20}$$

$I(t,c)$ рівне нулю, якщо *t* та *c* є незалежними.

Для того що виміряти користь терма в загальному просторі ознак, оцінку категорії по термам групують в два альтернативні шляхи:

$$I_{avg}(t) = \sum_{i=1}^{m} P_r(c_i) I(t,c_i), \tag{1.21}$$

$$I_{\max}(t) = \max_{i=1}^{m} \{I(t, c_i)\}. \quad (1.22)$$

Обчислення семантичної близькості слів засноване на наступній гіпотезі: якщо ознаки зустрічаються в одних і тих самих контекстах, то, вони, зазвичай, мають схожі значення. Слово може бути представлене вектором, компоненти якого отримані шляхом підрахунку числа входжень цього слова в різноманітних документах.

Тож виплив PMI на ймовірність появи будь-яких двох термів разом можна описати наступним чином: пари слів, які мають високий PMI майже з однаковою ймовірністю зустрічаються разом та окремо одне від одного; а пари, які мають низький PMI, зустрічаються в текстах окремо набагато частіше, а ніж разом. Таким чином PMI можна використовувати для: вилучення рідких термінів із текстів, аналізу емоційного забарвлення тощо.

Недоліком PMI є те що оцінка сильно залежить від граничних ймовірностей термів, як це видно в наступній еквівалентній формі:

$$I(t \mid c) = \log P_r(t \mid c) - \log P_r(t) \quad (1.23)$$

Для термів з однаковою умовною ймовірністю $P_r(t \mid c)$, рідкі терми матимуть більшу оцінку ніж загальні терми. Відповідно, оцінки не можна порівнювати за умови широко визначеної частоти.

Метод умовних випадкових полів (CRF). Цей алгоритм представляє собою неспрямовану ймовірнісну графічну модель, яка кодує умовні розподілення ймовірностей з заданим набором особливостей.

Нехай задана послідовність даних $X\{x_1, x_2, ..., x_n\}$ та відповідні ним маркери $Y\{y_1, y_2, ..., y_n\}$, CRF визначає умовну ймовірність наступним чином:

$$P(Y \mid X) = \frac{1}{z_x} \exp\left[\sum_i \sum_j \lambda_j f_j(y_{i-1}, y_i, X, i)\right], \quad (1.24)$$

де z_x – нормалізація, яка приводить суму всієї послідовності станів до одиниці; $f_j(y_{i-1}, y_i, X, i)$ – функція ознак;

j– відповідна вага ознаки f_j.

Основні переваги даного методу:

– при автоматичному виділенні сутностей та ключових слів показує кращі результати ніж MEMM (Марковські моделі із максимальною ентропією) [42] або HMM (Сховані Марковські моделі) [42];

– гнучкий у виборі ознак для навчання, при чому наявність умовної незалежності змінних не є обов'язковою умовою.

Основними недоліками CRF, по-перше, є обчислювальна складність аналізу навчаючої вибірки, що ускладнює постійне оновлення моделі при надходженні нових навчаючих даних; по-друге, при виділенні ключових слів не працює з невідомими словами (тобто зі словами, які не зустрічались в навчальній вибірці).

1.4 Огляд методів побудови та навчання класифікатора

В [9] методи побудови та навчання класифікатора поділяють на наступні групи:

– ймовірнісні: наївний баєсовський метод (NB)[44];

– лінійні: MaxEnt [2], метод опорних векторів (SVM)[44];

– метричні: метод k найближчих сусідів (k-NN) [46], Класифікатор Роші (Rocchio Classifier)[47];

– логічні: дерева рішень(DT)[48];

– методи на основі штучних нейронних мереж: FFBP, RNN, DAN2, CNN, ANN[49].

Класифікатор Роші. Даний класифікатор відноситься до моделей, які використовують векторний простір. Такі моделі базуються на гіпотезі суміжності. Ця гіпотеза полягає в наступному: документи в одному класі з суміжного регіону та регіони різних класів не перетинаються.

Класифікатор Роші розподіляє векторний простір ознак на регіони відносно центроїду (один для кожного класу), обчислений як центр мас всіх документів класу.

Для обчислення центроїду категорії, кожен текст з навчальної вибірки представляється в вигляді вектору $\overline{v(d)}$ спроектованого на словник навчальної вибірки. Зазвичай для визначення ваг ознак документа використовують метод TF-IDF, але можливе використання будь-яких інших методів. Тоді центроїд категорії розраховується за формулою:

$$\overline{\mu(c)} = \frac{1}{|D_c|}\sum_{d \in D_c} \overline{v(d)}, \tag{1.25}$$

де D_c – набір документів з навчальної вибірки, які належать класу *c*.

Межі між класами в класифікаторі Роші є набір точок, що знаходяться на однаковий відстані від центроїдів класів. Цей набір точок завжди є лінією. В загальному вигляді для M-вимірного простору, межа має вигляд гіпер-площини, яка визначається як набір точок $\bar{x}$:

$$\overline{w}^T \bar{x} = b, \tag{1.26}$$

де $\overline{w}$ – M-вимірний нормальний вектор гіперплощини, а *b*– константа. Наприклад, для двовимірного простору:

$$w_1 x_1 + w_2 x_2 = b. \tag{1.27}$$

Текст, який класифікується, буде віднесений до того класу, до центроїду якого він найближче знаходиться. Зазвичай для визначення відстані обчислюють косинус між векторами:

$$Assign\ d\ to\ c = \arg\max_{c'} \cos(\overline{\mu}(c'), \overline{v}(d)). \tag{1.28}$$

Класифікатор Роші простий та ефективний, але може бути не точним, якщо класи не є приблизно сферами зі схожими довжинами радіусів.

k-NN. Метод *k* найближчих сусідів – це метричний алгоритм, також як і класифікатор Роші заснований на оцінюванні схожості об'єктів. Але на відміну від класифікатора Роші, *k-nn* спирається не на відстань між центроїдом та текстом який класифікується, а на відстані між елементами класів. Тобто, об'єкт, який класифікується належить до того класу, до якого належить найближчий до нього об'єкт навчаючої вибірки. Нехай $X \in R^n$–множина об'єктів навчаючої вибірки, *Y*– множина допустимих класів, $\{(x_i, y_i)\}_{i-1}^{l}$– навчальна вибірка. Необхідно знайти множину класів $\{y_i\}_{i=1}^{m}$ для об'єктів $\{x_i\}_{i=1}^{m}$.

На множині об'єктів задається функція відстані, яка задається користувацькою функцією $\sigma(x, x')$. Частіше за все розглядаються наступні функції:

- Евклідова: $\sigma(x, x') = \sum_{i=1}^{n} (x_i - x_i')^2$;
- максимум модулів: $\sigma(x, x') = \max_i | x_i - x_i' |$;
- сума модулів: $\sigma(x, x') = \sum_{i=1}^{n} | x_i - x_i' |$.

Для довільного об'єкта *x* розташовують x_i в порядку збільшення відстані до *x*:

$$\sigma(x, x_{1,x}) \le \sigma(x, x_{2,x}) \le ... \le \sigma(x, x_{m,x}) \ , \qquad (1.29)$$

де $x_{i,x}$– об'єкт навчаючої вибірки, який є *i*-м сусідом об'єкта *x*. Аналогічне означення вводиться для класів на *i*-м сусіді $y_{i,x}$. Таким чином довільний об'єкт *x* формує свою перенумерацію вибірки. Загальна формула вибору найближчого сусіда має вигляд:

$$a(x) = \arg\max_{y \in Y} \sum_{i=1}^{m} \left[x_{i,x} = y\right] \omega(i, x), \qquad (1.30)$$

де $\omega(i, x)$– задана вагова функція, яка оцінює ступінь пливу *i*-го сусіда для класифікації об'єкта *x*.

Перевагами даного методу є:

– можливість оновлення навчальної вибірки без перенавчання класифікатора;

– відносно проста програмна реалізація;

– стійкість алгоритму до аномальних викидів у вихідних даних;

– легка інтерпретованість результатів роботи алгоритму;

– хороше навчання у разі випадку з лінійно нероздільними вибірками.

Недоліки k-NN:

– висока залежність результатів класифікації від обраної метрики;

– великий час роботи, через необхідність повного перебору навчаючої вибірки;

– неможливість вирішення задач великої розмірності за кількістю класів та документів.

NB. Метод Байєса полягає в наступному: нехай $P(c_i \mid d)$ – ймовірність того, що документ, представлений вектором $d = (t_1, t_2, ..., t_n)$, відповідає категорії $c_i, i = \overline{1, |C|}$. Таким чином класифікатор має знайти такі значення c_i та d, щоб $P(c_i \mid d)$ була максимальною:

$$CSV(d) = \arg\max_{c_i \in C} P(c_i \mid d) \tag{1.31}$$

$P(c_i \mid d)$ обчислюють використовуючи теорему Байєса:

$$P(c_i \mid d) = \frac{P(c_i) P(d \mid c_i)}{P(d)}, \tag{1.32}$$

де $P(c_i)$ – апріорна ймовірність того, що документ належить категорії c_i;

$P(d \mid c_i)$ – ймовірність знайти документ, представлений вектором $d = (t_1, t_2, ..., t_n)$, в категорії c_i;

$P(d)$ – ймовірність того, що довільно взятий документ можна представити у вигляді вектору ознак $d = (t_1, t_2, ..., t_n)$. Оскільки ознаки $t_1, t_2, ..., t_n$ задані заздалегідь,

та $P(d)$ не залежить від c_i, то дільник є константою, яка не впливає на вибір найбільшого зі значень.

Через велику кількість ознак $t_1, t_2, ..., t_n$ обчислення $P(d \mid c_i)$ досить складне, тому роблять "наївне" припущення, що будь-які 2 координати розглядаються як випадкові величини, статистично не залежать одна від одної, в такому випадку можна скористатися формулою:

$$P(d \mid c_i) = \prod_{k=1}^{n} P(t_k \mid c_i) \tag{1.33}$$

Далі всі ймовірності підраховуються за методом максимальної правдоподібності.

До переваг даного методу можна віднести:

– висока швидкість роботи;

– підтримка інкрементного навчання;

– відносно проста програмна реалізація;

– легка інтерпретованість результатів роботи алгоритму.

Недоліками NB є відносно низька якість класифікації та неможливість врахування залежності результату класифікації від поєднання ознак.

MaxEnt. Даний метод будує класифікаційну модель спираючись на ту саму інформацію, що й спрощений алгоритм Байєса: ці обидва алгоритми оперують таблицями ваг, які отримуються під час навчання з частотності класифікаційних ознак в межах класу, - по одному параметру на кожну пару класифікаційна ознака + клас. Тобто всього таких ваг $n*k$, де n – кількість класів, а k – кількість класифікаційних ознак. [2]

Відмінність же MaxEnt від алгоритму Байєса полягає в підході до пошуку ваг. Якщо спрощений алгоритм Байєса не враховує можливі залежності, то MaxEnt використовує поліноміальну логістичну регресію, щоб визначити категорію тексту, який передається.

Метод опорних векторів.[50] Метод опорних векторів є лінійним класифікатором, основна ідея якого полягає у переведенні початкових векторів у простір більш високої розмірності і пошук гіперплощини, що розділяє класи з максимальним зазором. Дві гіперплощини будуються по сторонам гіперплощини, що розділяє класи [50]. При чому вважається, що чим більша відстань від гіперпрощини, що розділяє класи до сформованих розділенням гіперплощин, тим менше буде середня похибка класифікатора. На рисунку 1.2 представлений приклад такого розбиття простору векторів документів.

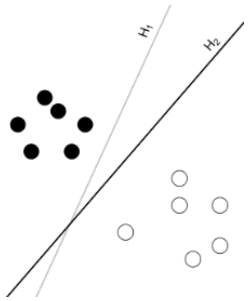

Рисунок 1.2 – Дві гіперплощини, H_1 та H_2, що розділяють класи, але тільки H_2 розділяє їх на максимальній відстані від кожної.

Нехай нам дана вибірка T, яка складається з n елементів:

$$T = ((v_i, y_i)|v_i \in R^k, v_i \in \{-1,1\}) , \quad (1.34)$$

де v_i – вектор ознак, розмірності k.

Необхідно знайти гіперплощину, яка б була максимально віддаленою від класів та розділяла їх наступним чином: $y_i = 1 = 1$ та $y_i = -1$. Будь-яку гіперплощину можна записати у вигляді множини точок:

$$W*x - b = 0 \quad (1.35)$$

Параметр $\frac{b}{||w||}$ визначає зміщення гіперплощини від початку координат вздовж нормалі w. Для того, щоб відстань між гіперплощинами була максимальною, треба вирішити наступну задачу оптимізації:

$$||w|| \to min,$$
$$y_i(w \cdot x_i - b) \geq 1 \text{ для } 0 \leq i \leq n \quad (1.36)$$

Класифікатор, побудований методом опорних векторів, є лінійним класифікатором, та зазвичай є стійким до перенавчання і добре масштабується.

Часто буває так, що знайти оптимальну гіперплощину неможливо, тоді класифікатор шукає гіперплощину, ітеративно мінімізуючи похибоки та максимізуючи відстань між класами.

Метод опорних векторів є ефективним методом класифікації текстової інформації і часто застосовується для визначення емоційного забарвлення текстової інформації. Його головним недоліком є те, що при пошуку гіперплощини, що розділяє класи, до уваги беруться лише граничні точки [4].

Поліноміальна логістична регресія. Дана модель представляє собою загальний випадок моделі логістичної регресії, в якій залежна змінна має більше 2-х категорій. Поліноміальна логістична регресія також як і метод опорних векторів відноситься до лінійних методів.

В поліноміальній логістичній регресії для кожної категорії залежних змінних будується рівняння бінарної логістичної регресії. При чому одна з категорій залежних змінних стає базовою, та всі інші категорії порівнюються з нею. Рівняння поліноміальної логістичної регресії прогнозує імовірність приналежності залежної змінної за значенням незалежних змінних.

Тобто щоб знайти оптимальні ваги для необхідно знайти набір параметрів, за яких досягається максимум наступної функції:

$$\log\ p(D,\gamma) = \sum_{(c,d)\in(C,D)} \log(p\langle d,\gamma\rangle) = \log \frac{\exp \sum_{i}^{n\times k} \gamma_i, f_i(c,d)}{\sum_{\widetilde{c}\in C} \exp \sum_{i}^{n\times k} \gamma_i, f_i(\widetilde{c},d)} \tag{1.37}$$

Зазвичай максимум такої задачі знаходять за допомогою градієнтного спуску.

1.5 Огляд методів оцінки ефективності класифікації

Основним критерієм при оцінці якості класифікації є комбінація точності і повноти.

Точність (precision) [11] класифікації в межах класу – це частка знайдених класифікатором документів, дійсно належать даному класу, щодо всіх документів, які система віднесла до цього класу.

Повнота (recall) [11] класифікації – це частка знайдених класифікатором документів, дійсно належать класу, щодо всіх документів цього класу в тестовій вибірці.

Для їх розрахунку окремо для кожної категорії (класу) будують таблиці контингентності (таблиця 1.1)[11].

Таблиця 1.1– Таблиця контингентності

Категорія i		Експертна оцінка	
		Позитивна	Негативна
Оцінка системи	Позитивна	TP	FP
	Негативна	FN	TN

Точність та повнота визначаються наступним чином:

$$preision = \frac{TP}{TP + FP} \tag{1.1}$$

$$recall = \frac{TP}{TP + FN} \tag{1.2}$$

На практиці для визначення точності та повноти часто використовують матриці неточностей (confusion matrix). Матриця неточностей – це матриця розмірності $N \times N$, де N – кількість класів, стовпці відповідають експертним рішення, а рядки – результатам роботи класифікатора. Під час класифікації документу з тестової вибірки, інкрементуємо число, яке стоїть на перетині рядка

класу, який був визначений класифікатором та стовпчику класу, до якого насправді відноситься документ. Загальна схема матриці неточностей наведена в таблиці 1.2

Таблиця 1.2– Матриця неточностей

	$recall$	$recall_1$	…	$recall_N$
$precision$		1	...	N
$precision_1$	1	$A_{1,1}$	…	$A_{1,N}$
…	…	…	…	…
$precision_N$	N	$A_{N,1}$	…	$A_{N,N}$

З такої матриці точність та повнота для кожного класу розраховується досить легко: точність дорівнює співвідношенню відповідного діагонального елементу матриці до суми усього рядка, а повнота – співвідношенню діагонального елементу до суми всього стовпчику:

$$preision_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{c,i}}, \tag{1.3}$$

$$recall_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{i,c}}, \tag{1.4}$$

Результуюча точність класифікатора розраховується як середнє арифметичне його точності по класам. Так само й повнота.

Для визначення загальної ефективності алгоритму часто використовують метрику F-міри, яка є гармонічним середнім між точністю та повнотою. F-міра розраховується за наступною формулою:

$$F = 2\frac{precision * recall}{precision + recall} \tag{1.5}$$

Така формула надає однакову вагу точності та повноті. Для надання переваги точності або повноті вводять додаткову константу:

$$F = (\beta^2 + 1)\frac{precision * recall}{\beta^2 precision + recall} \quad (1.6)$$

якщо β приймає значення в діапазоні $0<\beta<1$, то надається перевага точності, а якщо $\beta>1$– перевага надається повноті.

Як відзначено в [11], іноді для порівняння алгоритмів класифікації використовують специфічні характеристики, такі як BEP. BEP–величина, запозичена з економіки, що відображає обсяг виробництва і реалізації продукції, при якому витрати будуть компенсовані доходами, а при виробництві і реалізації кожної наступної одиниці продукції підприємство починає отримувати прибуток. В контексті даної задачі точка беззбитковості використовується як міра якості класифікації. Точка беззбитковості нарівні з F-мірою є збалансованою характеристикою точності і повноти.

Також при оцінці ефективності класифікації враховують швидкодію. Під швидкодією класифікатора розуміється час, що витрачається на віднесення документа до одного з класів. Що стосується завданням класифікації текстів швидкодію вимірюється як процесорний час (в секундах) або як кількість обчислювальних операцій, необхідне для класифікації. Вимірювання проводять на навчальній вибірці для оцінки швидкості процесу навчання і окремо на тестовій.

Ясно, що збільшення точності класифікації зазвичай призводить до зниження швидкодії через ускладнення вирішального правила, використовуваного в алгоритмі класифікації, а збільшення швидкодії супроводжується зниженням точності через спрощення роботи класифікатора.

1.6 Огляд аналогів на ринку схожих рішень

ApacheOpenNLP. Це бібліотека [51], яка представляє собою інструментарій на основі машинного для обробки текстів на природній мові. (Мова програмування – Java).

Вона підтримує найбільш поширені завдання обробки текстів на природній мові, такі як токенізація, сегментація пропозицій, частково-мовна розмітка, витяг змісту, розбір тексту, дозвіл кореферентності. Ці завдання, як правило, потрібні для створення більш складних методів по обробці тексту. Інструмент який виконую класифікацію текстів за заданими наперед категоріями має назву Document Categorizer (Doccat). Він заснований на описаному вище методі – MaxEnt. Для взаємодії з користувачем реалізований CLI інтерфейс. Для функції класифікації немає готових моделей, користувач сам їх формує за наявності достатньо великої навчальної вибірки.

В [2] було проведено практичне дослідження роботи ApacheOpenNLP та отримано, що навчання моделі з навчальною вибіркою в 9828 рядків тривало 1332 секунд, а F-міра класифікатора рівна 0.81.

uClassify [52] – безкоштовний веб-сервіс для автоматичної класифікації текстової інформації. На відміну від ApacheOpenNLP в uClassify є навчені моделі для класифікації текстів за деякими основними категоріями:

- визначення забарвлення тексту (позитивне чи негативне);
- визначення: текст написаний чоловіком чи жінкою;
- визначення мови тексту;
- розподіл між деякими з основних категорій (мистецтво, бізнес, комп'ютери, ігри, здоров'я, дім, відпочинок, наука, суспільство, спорт).

Є ще ряд інших функцій, але більшість з них доступна лише для англійської мови, деякі одна ще для французької, іспанської та швецької.

Carrot2[53] сортує текст та результати пошуку за категоріями. Ця система може згрупувати невеликі колекції документів, результати пошуку або короткий опис документів у відповідні тематичні категорії. Програма володіє відкритим вихідним кодом та дозволяє змінювати механізм групування. Дозволяє також використовувати готові компоненти для пошуку з різноманітних джерел, такі як GoogleAPI, APIBing, eToolsMetaSearch, Lucene, SOLR тощо.

KHCoder[53] – додаток для кількісного контент-аналізу, TextMining або корпусної лінгвістики. Може обробляти тексти японської, англійської, французької, німецької, італійської, португальської та іспанської мов. Для необроблених текстів, які вводяться, можна провести пошук та статистичний аналіз KWIC, статистику словосполучень, створити мережу зустрічностей, карту зв'язків слів, багатовимірне масштабування, кластерний аналіз та аналіз кореспонденції. KHCoder забезпечує різноманітні види функцій пошукового та статистичного аналізу, використовуючи засоби бекенду, такі як StanfordPOSTagger, Snowballstemmer, MySQL и R.

TAMSAnalyzer[53] – для OS X Macintosh є рішенням для розпізнання тем в текстах, таких як веб-сторінки, інтерв'ю, нотатки.

MaltParserта **LinkGrammar**[6] – синтаксичні парсери, які будують дерева залежностей для всіх речень тексту.

Висновок до розділу

При проведенні аналізу літературних джерел були оцінені основні складові розв'язання задачі класифікації неструктурованих текстових масивів інформації та їх методи. Виявлені основні проблеми, які потребують вирішення, а саме:

- зняття морфологічної та синтаксичної омонімії;
- позбавлення синтаксичної синонімії;
- вирішення кореферентності;

Метою дослідження є покращення якості моніторингу медіа активності шляхом розробки алгоритму автоматичної класифікації текстової інформації, що дозволяє підвищити точність та повноту аналізу даних.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати алгоритми та методи машинного навчання для вирішення задачі автоматичної класифікації текстів;

– обрати модель представлення текстової інформації в класифікаторі;

– розробити алгоритм попередньої обробки текстів відповідно до обраної моделі представлення текстової інформації;

– розробити модифікований метод класифікації текстової інформації;

– виконати програмну реалізацію розробленого алгоритму автоматичної класифікації текстової інформації;

– підготувати дані для оцінки якості класифікації;

– провести дослідження ефективності розробленої інформаційної технології.

2 РОЗРОБКА АЛГОРИТМУ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ТЕКСТОВИХ МАСИВІВ ІНФОРМАЦІЇ

2.1 Формальна постановка задачі класифікації текстів

Задача класифікації неструктурованих текстових масивів інформації полягає в наступному:

Існує множина документів $D = \{d_1,...,d_{|D|}\}$ та множина можливих категорій (класів) $C = \{c_1,...,c_{|C|}\}$. Є невідома функція $F : D \times C \to \{0,1\}$, яка для кожної пари <документ, категорія> визначає, чи відповідають вони один одному.

$$F(d_j, c_i) = \begin{cases} 1, \text{якщо } d_j \in c_i \\ 0, \text{якщо } d_j \notin c_i \end{cases} \quad (2.1)$$

Задача полягає в тому, що необхідно знайти максимально близьку до функції F функцію F'. F' називають класифікатором. [54]

Класифікацію розділяють на однозначну (якщо в задачі кожному документу $d \in D$, може відповідати лише одна категорія $c \in C$) та багатозначну (якщо кожному документу $d \in D$, може відповідати довільна кількість класів). [9]

2.2 Розробка модифікованого методу індексації документів

Як було сказано вище етап індексації документів для розв'язання задачі класифікації включає в себе токенізацію, лінералізацію та морфологічний розбір (нормалізацію).

Нехай на вхід подано деякий текст d. В якості токену оберемо речення – $s_i \in d, i = \overline{1,n}$, де n– кількість речень в тексті d. Кожному слову $w_{ij} \in s_i$ проставимо у відповідність *tag* – граматичний клас слова. Цей етап ускладняється наявністю омонімії. Англійській мові притаманна лише частиномовна омонімія, тобто омонімія, за якої одне й те ж саме слово відноситься до різних частин мови. Тому

в *tag* слова будемо виносити лише частину мови слова, яке розглядається. Таким чином *tag* може приймати одне зі значень множини $\{N:noun, Adj:adjective, P:pronoun, Num:numeral, V:verb, Adv:aderb, S:stative\}$. Stative – слова, які означають психічний або фізичний стан, часто з модальним забарвленням. В реченні вони виступають в ролі присудка безособового речення. Наприклад, *alive, pity, afraid*. В множину можливих значень не винесено службові частини мови (*conjunctions, prepositions, practices, modal verbs etc*.) тому що вони не несуть змістовного навантаження і після виконання морфологічного розбору будуть видалені із множини ознак.

Визначення тегів однозначних слів є тривіальною задачею, а для визначення тегів слів-омонімів застосуємо гібридний метод позбавлення омонімії на основі умовних ймовірностей та продукційних правил.

Ймовірність того, що слово з тегом X буде слідувати за словом з тегом A можна розрахувати за формулою:

$$P(X \mid A) = \frac{C(A,X)}{C(A)}, \tag{2.2}$$

Де $C(A,X)$– число випадків, коли слова з переліченими тегами слідують один за одним у відповідному порядку. Формула, яка враховує наступне слово:

$$P'(X \mid A) = \frac{C(X,A)}{C(A)}. \tag{2.3}$$

Результуюча формула ймовірності кожного тега x приймає в розрахунок як попереднє так і наступне слово:

$$\overline{P}(x \mid A,B) = \sum_{a \in A} P(x \mid a) * \sum_{b \in B} P'(x \mid b), \tag{2.4}$$

де A та B – множини допустимих тегів для сусідніх слів зліва та справа відповідно.

Для 3-грам, в свою чергу, використовують наступну формулу:

$$\overline{P}(x \mid A,B,Y,Z) = \sum_{\substack{a \in A \\ b \in B}} P(x \mid a,b) * \sum_{\substack{b \in B \\ y \in Y}} P'(x \mid b,y) * \sum_{\substack{y \in Y \\ z \in Z}} P'(x \mid y,z) \tag{2.5}$$

$$P'(x \mid b, y) = \frac{C(b, x, y)}{\sum_{t \in T} C(b, t, y)}, \tag{2.6}$$

де T – множина всіх тегів.

Кожному неоднозначному слову формується список допустимих тегів, після чого кожен тег отримує оцінку ймовірності по формулам (2.4) або (2.5). По завершенню оцінок словам співставляють теги з найбільшою ймовірністю[55].

Тобто для кожного слова речення необхідно вибрати тег таким чином щоб максимізувати функцію[56]:

$$\arg\max_t P(w \mid t) * P(t \mid c^n), \tag{2.7}$$

де w – поточне слово;

t – тег;

c^n – ланцюжок тегів від першого слова речення, до останнього розібраного, або визначеної за умови довжини n.

$P(t \mid c^n)$ – умовна ймовірність появи даного тега t, за умови, що попередні n тегів визначені.

$P(w \mid t)$ – умовна ймовірність появи в поточному місці слова w, за умови, що це слово має граматичний клас t.

На таких правилах заснований HMM. Він реалізується методом динамічного програмування Вітербі. Часто для спрощення обчислень вдаються до обмежень розмірностей послідовностей тегів.

В якості детермінованих правил зняття омонімії буде використаний словник контекстів виду <*омонім, елемент контексту, лема, тег*> [56]. Лема – початкова форма слова. Тобто якщо виникне подія появи омоніму поряд з відповідним елементом контексту, то такий омонім буде замінений відповідною лемою. Звісно елементи контексту сильніше або слабкіше впливають на вибір значення омоніма в залежності від їх розміщення відносно омоніма, а при рівних позиціях – від співвідношення ймовірностей елементарних подій, які відносяться до різних лем.

Наприклад, елемент контексту, який надає лему №1 з ймовірністю 0.85, а лему №2 з ймовірністю 0.15, "сильніший" за елемент зі співвідношенням ймовірностей 0.65 до 0.35, а той в свою чергу "сильніший" за елемент контексту, який обидві леми обирає з ймовірністю 0.5. Точно так, якщо в деякій позиції співвідношення математичного очікування вибору одної з двох лем, обчислених по всім елементам контексту, які знаходяться в даній позиції складає 0.8 та 0.2, а в іншій – 0.6 та 0.4, то перша "позиція" буде сильнішою за другу, оскільки в неї більш чітко виражена перевага.

В якості функції обчислення ступеню впливу елемента контексту на вибір леми можна використовувати наступну формулу ентропії:

$$F(p,q) = 1 + p * \log_2 p + q * \log_2 q, \tag{2.8}$$

де p та q – ймовірності, такі що $p + q = 1$. Головна властивість даної функції полягає в тому, що чим ближче p та q один до одного, тим менше значення функції $F(p,q)$ і навпаки.

Отже, отримаємо наступний алгоритм індексації текстових документів.

Введемо наступні позначення:

$D = \{d_1,...,d_n\}$ – множина документів навчальної вибірки;

$S^i = \{s_1^i,...,s^i{}_{|d_i|}\}$ – множина речень документа d_i;

$s_j^i = \{x_1,...,x_m\}$, $i = \overline{1,n}$, $j = \overline{1,|s^i|}$ – множина слів речення s_j^i.

* – початок речення;

$STOP$ – кінець речення;

T – множина всіх можливих тегів.

$q(t \mid u,v)$ – ймовірність переходу в стан t при попередніх станах (u,v);

$e(x \mid t)$ – ймовірність присвоєння слову x тегу $t \in T$;

$R = \{r_1,...,r_m\}$ – множина лематизованих(стематизованих) слів;

$Y = \{y_1,...,y_m\}$ – послідовність тегів наданих термам речення;

$STEM(x)$ – функція стемінгу Портера;

W^o – словник контекстів;

H – множина контекстів;

L – множина лем, які може приймати омонім;

$g(h)$ – значення сили елементу контексту;

$p(l,h)$ – ймовірність породження леми l контекстом h.

Визначення: $T_{-1} = T_0 = \{*\}$, $T_k = T$ для $k \in 1...m$.

Модифікований алгоритм попередньої обробки та індексації документів буде виглядати наступним чином:

for $k = 1....m$

if $x_k \in W^o$

for $h \in H$, $l \in L$

$g(h) = 1 + p(l,h) * \log_2 p(l,h)$

$P(l) \leftarrow \sum_h p(l,h) * g(h)$

$r_k \leftarrow \arg\max_l P(l)$

$y_k \leftarrow \arg\max_{t:l} P(l)$

else for $u \in T_{k-1}, v \in T_k$

$\pi(k,u,v) \leftarrow \max_{w \in T_{k-2}} (\pi(k-1,w,u) \times q(v \mid w,u) \times e(x_k \mid v))$

$bp(k,u,v) \leftarrow \arg\max_{w \in T_{k-2}} (\pi(k-1,w,u) \times q(v \mid w,u) \times e(x_k \mid v)$

$r_k \leftarrow STEM(x_k)$

$(y_{n-1}, y_n) \leftarrow \arg\max_{u,v} (\pi(n,u,v) \times q(STOP \mid u,v))$

$W^o add\ (x_k, u, v,)$

for $k = (n-2)....1$

$y_k \leftarrow bp(k+2, y_{k+1}, y_{k+2})$

$W^o add\ (x_k, bp(k,u,v), r_k, y_k)$

Для того щоб зменшити час роботи алгоритму Вітербі, який в загальному вигляді рівний $O(n|T|^3)$, оскільки виконується обчислення $q(v|w,u)\times e(x_k|v)$ для всіх можливих k,t,u,v, пропонується застосовувати перевірку на наявність контекстних правил для поточного слова x_k. Якщо такі правила є, для кожної пари <омонім, елемент контексту> із словника контекстів обирається лема та ймовірність її породження даним елементом контексту. Після чого для кожної леми обчислюється ймовірностей, помножена на значення "сили" елемента (обчисленої за формулою (2.8)). Значенням омоніма в такому разі вважається лема з найбільшою зваженою сумарною ймовірністю.

І тільки в разі відсутності правил в контекстному словнику позбавлення омонімії, буде виконуватися ітерація алгоритму Вітербі. Отримане правило після його відпрацювання буде додане до бази правил контекстних елементів W^o для можливості його застосування на подальших ітераціях, тобто виконується процес самонавчання моделі.

2.3 Загальна модель рішення поставленої задачі класифікації

В розділі 1.1 вказано, що задача класифікації складається з 4-х етапів. На першому етапі попередньої обробки та індексації документів буде застосований гібридний алгоритм попередньої обробки документів, розроблений в розділі 2.2. В [1] представлений порівняльний аналіз алгоритмів визначення ваг ознак документів, за яким найкращі показники отримані алгоритмом TF-SLF. Тому на етапі 2 – визначення ваг ознак документу буде застосований саме він. Для етапу 3, в якості методу побудови та навчання класифікатора, буде використаний класифікатор Роші, який дає непогані результати та є досить простим в реалізації. Та 4-й етап оцінки якості класифікації буде проведений шляхом розрахунку F-міри.

Тож загальна схема класифікації має наступний вигляд:

Крок 1: Розбиття документу $d_i \in D$ на речення $S^i = \{s_1^i,...,s^i{}_{|d_i|}\}$.

Крок 2: Розбиття речення s_j^i на масив слів $\{x_1,...,x_m\}$, $i = \overline{1,n}$, $j = \overline{1,|s^i|}$.

Крок 3: Індексування $\forall s_j^i$ алгоритмом, представленим в розділі 2.2.

Крок 4: Зважування термів за формулою (1.16).

Крок 5: Формування множини ознак, з урахуванням порогової ваги ознак β.

Крок 6: Розрахунок центроїду категорії за формулою (1.25)

Крок 7: Виконання кроків 2 – 4 для тексту, який класифікується – d^t.

Крок 8: Визначення мінімальної відстані між $\overline{v(d^t)}$ та центроїдами класів $c_i, i = \overline{1,n}$ за формулою (1.28).

Висновок до розділу

В даному розділі наведено формальну постановку досліджуваної задачі автоматичної класифікації текстових масивів інформації.

Розроблено гібридний алгоритм індексації неструктурованих текстових масивів інформації. Даний алгоритм присвоює кожному терму відомість про частину мови слова у вигляді тегу. Для позбавлення частиномовної омонімії англомовних текстів застосовані статистичний метод Вітербі та детерміновані правила, засновані на словнику контекстів.

Даний алгоритм індексації включений в загальну модель класифікації текстів на основі TF-SLF та класифікатора Роші.

Результати розділу були опубліковані в [1,3].

3 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Інформаційне забезпечення

3.1.1 Вхідні дані

Вхідними даними є:

- множина допустимих тегів слів;
- корпус з морфологічною розміткою та знятою омонімією;
- набір контекстних правил для зняття омонімії;
- навчальна вибірка;
- тестова вибірка.

Прикладом множини допустимих тегів для англомовних текстів наведений в таблиці 3.1.

Таблиця 3.1– Множина допустимих тегів текстів

N	*Noun (іменник)*
Adj	*Adjective(прикметник)*
P	*Pronoun (займенник)*
Num	*Numeral (числівник)*
V	*Verb (дієслово)*
Adv	*Adverb (прислівник)*
S	*Stative (слова з модальним забарвленням – присудки в безособовому реченні)*

Корпус з морфологічною розміткою та знятою омонімією міститься в машино зчитуваному форматі з розширенням .json. Нижче приведений фрагмент цього JSON файлу.

```
{
    {
    "tag": "N",
    "word": "control"
    },
    {
    "tag": "P",
    "word": "that"
    },
    {
    "tag": "V",
    "word": "have"
    },
    {
    "tag": "Num",
    "word": "two"
    },
    {
    "tag": "Adj",
    "word": "more"
    },
    {
    "tag": "N",
    "word": "side"
    }
},
{
    {
    "tag": "P",
    "word": "each"
    },
    {
    "tag": "V",
    "word": "separate"
    },
    {
    "tag": "N",
    "word": "divider"
    },
    {
    "tag": "V",
    "word": "can"
    },
    {
    "tag": "V",
    "word": "be"
    },
    {
    "tag": "V",
    "word": "drag"
```

```
        },
        {
        "tag": "N",
        "word": "user"
        }
}
```


В якості навчальної вибірки був відібраний корпус з моніторингового проекту «ОКО»[57]. Навчальна вибірка надається в форматі .txt, в якому текст та категорія розміщені в один рядок та категорія від тексту відокремлена «@». На рисунку 3.1 наведено фрагмент навчальної вибірки.

Рисунок 3.1 – Фрагмент файлу навчальної вибірки

Тестова вибірка також відібрана з проекту «ОКО» і надається в форматі .txt. Кожен текст тестової вибірки має починатися з нового рядка. На рисунку 3.2 наведено фрагмент тестової вибірки.

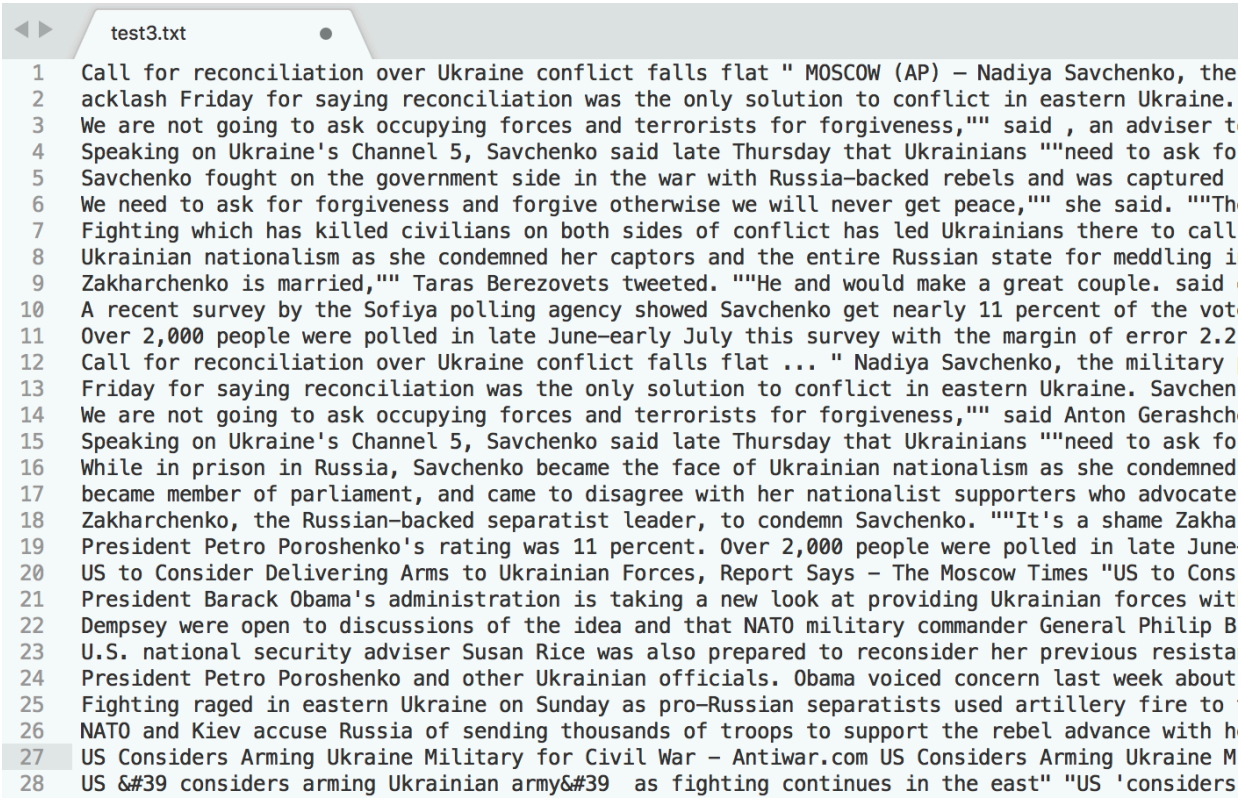

Рисунок 3.2 – Фрагмент файлу тестової вибірки

3.1.2 Вихідні дані

Вихідними даними є:

– класифікаційна модель;

– час створення моделі;

– час роботи класифікатора;

– прокласифіковані тексти;

Класифікаційна модель міститься в форматі .bin та може бути використана повторно для різних тестових даних.

Час створення моделі та час роботи класифікатора зберігаються для оцінки ефективності роботи класифікатора.

Прокласифіковані тексти зберігаються в форматі .csv. На рисунку 3.3 приведено фрагмент результатів роботи класифікатора.

1	NotrelatedtoUkraine	In the past decade, the path from defiant outsider to success as an insider has grown astonishingly short. Although politics provides the most glaring
2	Sport	Back in March, during the fifty-ninth minute of a Champions League match between Borussia Dortmund and Benfica, the eighteen-year-old America
3	NotrelatedtoUkraine	Decades before hip-hop was born, in New York, swing dancers (like the couple pictured above) favored a hop called the Lindy. On June 19 and June 27
4	Other	Mike Scheidt is the guitarist and vocalist behind YOB, which practices an extreme style of rock called doom metal. The music is bleak and slow, and te
5	NotrelatedtoUkraine	According to the chef Claire Welle—who closed her much loved Clinton Hill café Tilda All Day in December, after an owner dispute, and
6	NotrelatedtoUkraine	On a balmy afternoon in the spring of 1966, three young men from the Mattachine Society, an early gay-rights organization, entered Julius’
7	FightinginEasternUkraine	A great work by Darius Milhaud (1892-1974) is like a blast of complicated sunshine. The composer, a scion of one of the most ancient and distinguishe
8	FightinginEasternUkraine	On the morning of June 7th, several spectators gathered by the side of a narrow country road in Ballig, on the Isle of Man, to witness the third full da
9	NotrelatedtoUkraine	What happens in the Trump era when art and politics collide? Rebecca Mead joins Dorothy Wickenden to discuss the controversy over a production c
10	Ukrainianeconomy/industry	advertisement <a href="http://www.newyorker.com/humor/daily-shouts/technology-to-optimize-your-workday?mbid=rss-apple-news">See the
11	NotrelatedtoUkraine	After consulting a crack team of experts, Colin and I determined that we do not have any tattoos (and that Colin does have a fair number of moles). E
12	NotrelatedtoUkraine	On Wednesday, as expected, the Federal Reserve raised short-term interest rates by a quarter of a percentage point, to 1.25 per cent. This is still a v
13	NotrelatedtoUkraine	The epithet has the upper hand. While the highest stage of celebrity may be recognition by first name, the epithet has achieved a new standard: rec
14	FightinginEasternUkraine	In 1978, Frank Barrett, a construction worker in Dawson City, a town in the northern Yukon, was excavating the site of a new recreation center wher
15	Ukrainianeconomy/industry	“For those commuters with more skeptical bosses, here’s a tip: New Jersey Transit and the M.T.A. offer delay notes, similar to note
16	NotrelatedtoUkraine	Earlier this month, the Philadelphia rock-and-roll band the War on Drugs announced the follow-up to 2014’s “Lost in the Dream,&#
17	FightinginEasternUkraine	<figure class="block-container embed image"><img src="https://media.newyorker.com/photos/5942a87
18	NotrelatedtoUkraine	On Monday, TVRain, the only private television outlet in Russia whose political coverage is not guided by loyalty to the Kremlin, showed a split screen
19	NotrelatedtoUkraine	The story of how Nricco Iseppi, a master falconer, came to Riot Games has, among the company’s staff, acquired the malleability of myth. A
20	NotrelatedtoUkraine	Within hours of the shooting of the House Majority Whip, Steve Scalise, and four others, one couldn’t help but feel tired watching the predic
21	FightinginEasternUkraine	As the jury entered its third day of deliberations in Bill Cosby’s criminal trial for the alleged sexual assault of Andrea Constand, in 2004, I four
22	FightinginEasternUkraine	On the morning of December 14, 2012, when a gunman opened fire inside Sandy Hook Elementary School, in Newtown, Connecticut, I was in eastern
23	NotrelatedtoUkraine	The Congressional Baseball Game takes place every June, and legislators prepare for it diligently. For weeks leading up to the game, they attend regu
24	NotrelatedtoUkraine	While entering the marble lobby of Trump Tower, on Fifth Avenue, on a recent Friday afternoon, visitors walked past helmeted police officers holding
25	MH-17crash	advertisement <a href="http://www.newyorker.com/humor/daily-shouts/birthday-cards-for-donald-trump?mbid=rss-apple-news">See the rest
26	NotrelatedtoUkraine	When Clarissa Dalloway thinks that it’s “very, very dangerous to live even one day,” what, exactly, does she have in mind?
27	NotrelatedtoUkraine	If there was an organizing principle in Attorney General Jeff Sessions’s testimony before the Senate Intelligence Committee on Tuesday&#x
28	NotrelatedtoUkraine	I first encountered the work of the British traveller, archeologist, and spy Gertrude Bell many years ago, while hunting in the archives for a Carmelite
29	FightinginEasternUkraine	<a href="http://www.condenaststore.com/~/cartoon-bank?utm_medium=referral&utm_source=NewYorker&utm_content=daily-cartoo
30	NotrelatedtoUkraine	Donald Trump is not, by all accounts, a great reader. But he's memorized the Roy Cohn playbook, and in his first Cabinet meeting, on Monday,
31	MH-17crash	WASHINGTON (The Borowitz Report)—An Alaba

Рисунок 3.3 – Фрагмент .csv файлу з результатами класифікації текстів

3.2 Програмне та технічне забезпечення

3.2.1 Засоби розробки

Для програмної реалізації алгоритму автоматичної класифікації текстової інформації використано мову Java[59], платформа для створення графічного інтерфейсу JavaFX[60], концепція розподілу даних програмного забезпечення MVC[61] та інтегроване середовище розробки Eclipse IDE [62].

JavaFX — на основі Java для створення додатків з графічним інфтерфейсом, яка може бути використана як для створення настільних додатків так і для веб-додатків та мобільних додатків. До основних переваг JavaFX можна віднести:

- дозволяє створювати додатки з сучасним GUI;
- кросплатформеність;
- присутність менеджерів компоновки;
- можливість використання css стилів.

Для JavaFX створений засіб візуальної розмітки JavaFX Scene Builder [63], який дозволяє швидше та простіше створювати GUI.

Шаблон MVC забезпечує покращення розділення відповідальності. Тобто, Model відповідає за роботу з даними, формує їх структуру, представлений java класами. View формує зовнішній вигляд екранної форми додатку, описаній в файлах з розширенням .fxml. Controllerспівставляє дані Model до відповідного View, обробляє події та ініціює роботу View.

Eclipse підтримує інструменти (у вигляді плагінів) для проведення тестування, системи контролю версій (наприклад, Git), засоби автоматизації роботи з програмними проектами.

3.2.2 Вимоги до технічного забезпечення

Даний програмний продукт являє собою настільний інформаційну технологію з автоматичної класифікації текстової інформації.

Для оптимальної роботи ПЗ у користувача необхідна наявність наступних технічних засобів:

а) комп’ютер з такою конфігурацією:

1) процесор з тактовою частотою не менше ніж 1 ГГц;

2) об’эм оперативної пам’яті не менше 518 МБ;

3) інші складові можутьмати будь-які параметри, тому що вони не значним чином впливають на роботу системи.

б) додатково має бути встановлене таке програмнезабезпечення:

1) операційна система - будь-яка з Windows 10/8.1/8/7/Vista/ 32-bit/64-bit, Mac OS X 10.6 абоновішіверсії, ОС Linux;

2) JDK 8.0;

3.2.3 Архітектура програмного забезпечення

3.2.3.1 Діаграма класів

У частині графічного матеріалу представлена схема структурна класів ПЗ, які відповідають за виконання таких функцій, як розбір JSON файлу з контексними правилами та вибіркою промаркерованих текстів, позбавлених омонімії, створення усіх об’єктів класифікатора (словника, категорії, набору документів, слова), збереження даних процесу створення моделі та класифікації, стематизація слів збереження вихідних даних в CSV файл, запуск класифікатора, зчитування та обробка вхідних даних (навчальної та тестової вибірок), доповнення контекстних правил, відображення графічного інтерфейсу користувача, візуалізації отриманих результатів класифікації.

Схема містить 18 класів, а саме:

– MainApp – головний клас, що ініціює запуск усіх інших класів;

– ParseConfig – клас, що відповідає за розбір JSON файлу з контексними правилами та вибіркою промаркерованих текстів, позбавлених омонімії;

– Classific – клас, що реалізує початкові кроки класифікації;

– Category– клас-сутність категорії;

– Document –клас-сутність документу;

– Word– клас-сутність слова;

– Indexing – клас, що реалізує алгоритм попередньої обробки текстів;

– ContextRule– клас-сутність контекстних правил;

– ContextRulesIndexing – клас, що реалізує алгоритм тегування слів на основі контекстних правил;

– StaticIndexing– клас, що реалізує алгоритм Вітербі;

– Stemmer– клас, що реалізує алгоритм стемінгу Портера для англомовних текстів;

– TF_SFL – клас, що реалізує алгоритм обчислення TF-SLF для векторів документів та центроїдів категорій;

– Vocabular–клас, що формує множину ознак документів;

– ClModel – клас, що реалізує представлення та збереження класифікаційної моделі;

– ModelsOverviewController – клас, щозабезпечує передачу даних на viewClassified;

– Predict – клас, який реалізує обчислення найближчої категорію до тексту, який класифікується за класифікатором Роші;

– ClTexts – клас,що реалізує функції представлення та збереження класифікованих текстів;

– ClassifiedController – клас, що забезпечую передачу даних на viewClassified.

3.2.3.2 Специфікація функцій

Специфікація функцій наведена в таблицях 3.2 – 3.17.

Таблиця 3.2 – Специфікація функцій класу MainApp

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
getModelData	Повертає набір моделей	modelData	-	
getTextData	Повертає набір прокласифікованих текстів	textData	-	
start	Головний метод запуску JavaFX	-	Stage primaryStage	Основний контейнер View
initRootLayout	Ініціює відображення кореневого макету	-	-	-
showCreateModel	Ініціює відображення макету створення моделі	-	-	-
showClassifiedTexts	Ініціює відображення прокласифікованих текстів	-	-	-
getPrimaryStage	Повертає головне View	-	-	-
main	Головний метод програми	-	String[] args	Аргументи запуску програми
getAllCategories	Повертає список категорій	String[]	List<String> corpus	Корпус текстів

Таблиця 3.3 – Специфікація функцій класу ClModel

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
ClModel	Конструктор моделі	-	String trainPath	Шлях до навчальної вибірки
getName	Повертає ім’я моделі	String name.get()	-	-
setName	Задає ім’я моделі	-	String name	Ім’я моделі
nameProperty	Повертає ім’я моделі у типі StringProperty	StringProperty name		
getTrainPath	Повертає шлях до навчальної вибірки	StringtrainPath.get()	-	-
setTrainPath	Задає шлях до навчальної вибірки	-	String trPath	Шлях до навчальної вибірки
trainPathProperty	Повертає шлях до навчальної вибірки у типі StringProperty	StringProperty trainPath	-	-
getSavePath	Повертає шлях до збереженної моделі	String savePath.get()	-	-
setSavePath	Задає шлях збереження моделі	-	String svPath	Шлях для збереження моделі

Продовження таблиці 3.3

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
savePathProperty	Повертає шлях збереження моделі у типі StringProperty	StringProperty savePath	-	-
getCreateDate	Повертає дату створення моделі	LocalDate createDate.get()	-	-
setCreateDate	Задає дату створення моделі	-	LocalDate createDate	Дата створення моделі
createDayProperty	Повертає дату створення моделі у типі ObjectProperty <LocalDate>	Object Property <LocalDate> createDay	-	-

Таблиця 3.4 – Специфікація функцій класу ClassifiedController

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
initialize	Ініціалізує таблиці з прокласифікованими текстами	-		
setMainApp	Додає дані в таблицю	-	MainAppmainApp	Посилання на головний клас

Таблиця 3.5 – Специфікація функцій класу ClTexts

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
ClTexts	Конструктор текстів		String text	Текст документу
getCategory	Повертає категорію тексту	String category.get()	-	-
setCategory	Задає категорію	-	String category	Категорія тексту
categoryProperty	Повертає назву категорії тексту у типі StringProperty	StringProperty category	-	-
getText	Повертає текст	String text.get()	-	-
setText	Задає текст		String text	Текст документу
textProperty	Повертає текст у типі StringProperty	StringProperty text	-	-

Таблиця 3.6 – Специфікація функцій класу HistoryOverviewController

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
initialize	Ініціалізує таблицю зі списком моделей	-	-	-
setMainApp	Додає дані про моделі в таблицю	-	MainAppmainApp	Посилання на головний клас

Таблиця 3.7 – Специфікація функцій класу Classific

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
readTrain	Зчитує навчальну вибірку	List<String>	String path	Шлях до навчальної вибірки
readTest	Зчитує тестову вибірку	List<String>	String Path	Шлях до тестової вибірки
getTests	Повертає тестову вибірку	List<String>	-	-
getTrains	Повертає навчальну вибірку	List<String>	-	-
getTestText	Повертає текст з тестової вибірки за заданим індексом	String	int index	Індекс тексту в тестовій вибірці

Таблиця 3.8 – Специфікація функцій класу Indexing

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
isStopWord	Перевіряє чи є поточне слово стоп словом	boolean	String w	Поточне слово
getTag	Визначає тег слова	String	int index, String[] sentense	Індекс поточного слова, речення

Таблиця 3.9 – Специфікація функцій класу ParseConfig

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
readStaticInd	Зчитує промаркеровану вибірку, позбавлену омонімії	Word[]	String path	Шлях до .json файлу з промаркерованою вибіркою, позбавленою омонімії
readContextRules	Зчитує контекстні правила	ContextRule[]	String path	Шлях до .json файлу з контекстними правилами

Таблиця 3.10 – Специфікація функцій класу StaticIndexing

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
calcTag	Визначає тег слова за алгоритмом Вітербі	String	String word	Поточне слово

Таблиця 3.11 – Специфікація функцій класу ContextRulesIndexing

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
calcTag	Визначає тег слова за словником контекстних правил	String	Stringword, String context1, String context2, String context3	Поточне слово, попереднє слово вид поточного, слово на відстані 2 від поточного, слово на відстані 3 від поточного

Таблиця 3.12 – Специфікація функцій класу ContextRule

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
getContext1	Повертає контекст для на відстані 1	Word	-	-
getContext2	Повертає контекст для на відстані 2	Word	-	-
getContext3	Повертає контекст для на відстані 3	Word	-	-
getWord	Повертає слово, для якого прописане поточне контекстне правило	String	-	-
getTag	Повертає тег для поточного слова	String	-	-
createNewRule(): void	Створює нове контексне правило	-	String context, String word, String tag	Контекст для поточного слова, поточне слово, тег

Таблиця 3.13 – Специфікація функцій класу Stemmer

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
Stemmer	Конструктор класу (встановлює початкові параметри)	-	-	-
vowelinstem	Перевіряє чи містить послідовність *0…j* голосну	boolean	-	-
doublec	Перевіряє наявність подвоєння приголосних	boolean	int j	Індекс символа
toString	Конвертує стематизоване слово в тип *String*	String	-	-
getResultLength	Повертає довжину стематизованого слова	Int	-	-
getResultBuffer	Повертає стематизоване слово в типі *Char[]*	char[]	-	-
cons	Перевіряє чи є символ з індексом*i*приголосним	boolean	inti	Індекс літери в слові
m	Вимірює кількість приголосних, які ідуть підряд	int	-	-

Продовження таблиці 3.13

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
cvc	Перевіряє наявність форми *consonant - vowel - consonant*	boolean	inti	Індекс символа
ends	Перевіряє закінчення слова	boolean	String s	Слово, яке стематизується
setto	Встановлює (j+1),...k символам с слові s	-	String s	Слово, яке стематизується
r	Якщо кількість приголосних, які йдуть підряд більше нуля викликає фукнцію *setto*	-	String s	Слово, яке стематизується
step1	Перший крок стемінгу: позбаляє множинної форми слова, закінчення *-ed* або *-ing* тощо.	-	-	-
step2	Другий крок: перетворює *y* у *i* , коли є ще одна голосна в корні	-	-	-
step3	Третій крок: позбавляє подвійних суфіксів	-	-	-
step4	Четвертий крок: опрацьовує закнчення*-ic-, -full, -ness*	-	-	-
step5	П’ятий крок: видаляє *-ant, -ence* тощо, в конструкціях <c>vcvc<v>	-	-	-
step6	Шостий крок: видаляє *-e*.	-	-	-
stem	Функція, яка об’єднує всі етапи стематизації.	-	-	-

Продовження таблиці 3.13

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
step1	Перший крок стемінгу: позбаляє множинної форми слова, закінчення *-ed* або *-ing* тощо.	-	-	-
step2	Другий крок: перетворює *y* у *i* , коли є ще одна голосна в корні	-	-	-
step3	Третій крок: позбавляє подвійних суфіксів	-	-	-
step4	Четвертий крок: опрацьовує закнчення- *ic-, -full, -ness*	-	-	-
step5	П’ятий крок: видаляє - *ant, -ence* тощо, в конструкціях <c>vcvc<v>	-	-	-
step6	Шостий крок: видаляє - *e*.	-	-	-
stem	Функція, яка об’єднує всі етапи стематизації.	-	-	-

Таблиця 3.14 – Специфікація функцій класу Category

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
getName	Повертає назву категорії	String	-	-
setName	Встановлює назву категорії	-	String name	Назва категорії
getCentrSLF	Повертає SLF центроїду	double[]	-	-
setCentrSLF	Встановлює SLF центроїду	-	double[] c	Вектор SLF
setA	Встановлює параметр *a*	-	double a	Парамерт a
category	Конструктор категорії	-	String n, document[] d, List<String>voc	Назва категорії, перелік документів, які належать цій категорії, словник
calcCentrSLF	Обчислює SLF центроїду категорії	double[]	int size, document[] c_d, document[] n_cd	Розмір SLF,документи, які належать до цієї категорії, документи, які не належать до неї

Таблиця 3.15 – Специфікація функцій класу Document

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
getCat	Повертає назву категорії	String	-	-
setCat	Задає категорію тексту	-	String nameCat	Назва категорії
getWords	Повертає слова тексту	word[]	-	-
setWords	Задає слова тексту	-	word[] words	Слова тексту
setVectorSLF	Задає SLF вектор тексту	-	double[] v	Вектор SLF
getVectorSLF	Повертає вектор SLF	double[]	-	-
document	Конструктор документу	-	-	-
document	Конструктор документу	-	List<String>textes, intidDoc	Перелік документів, індекс документу
calcWords	Формує масив важливих слів, нормалізованих слів тексту	word[]	String doc	текст
removePunctuation	Видаляє пунктуацію та зайві символи	String	String str	Речення тексту

Таблиця 3.16 – Специфікація функцій класу TF_SLF

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
getSLF	Повертає вектор SLF	double[]	-	-
setSLF	Задає вектор SLF		double[] slf	Вектор SLF
tf_slf	Конструктор	-	document[] docs, List<word[]>voc, String[] cats	Масив документ, словник слів, список категорій
tfslf	Обчислює tf-slf для одного тексту по заданому SLF	double[]	String[] text, double[] SLF, List<word[] >voc	Текст, для якого обчислюється TF-SLF, вектор SLF, словник
calcSLFList	ОбичслюєSLFдля заданої множини текстів	-	document[] docs, List<word[] >voc, String[] cats	Масив документів, словник, список категорій
calcTF	Обчислює TF для поточного тексту	double[]	String[] text, List<word[]>voc	Текст, для якого обчислюється TF, словник
normalizeTFSLF	Нормалізує вектор TF-SLF	double[]	double[] tfslf	Вектор TF-SLF

Таблиця 3.17 – Специфікація функцій класу Predict

Назва	Опис	Повертає результат	Перелік параметрів	Семантика параметрів
calcCategor	Визначає категорію для тексту, який класифікується	category	String text, tf_slf t, List<word[]>voc, category[] cats	Текст, якій класифікується; tf-slf тексту, який класифікується; словник; набір доступних категорій
calcCos	Обчислює відстань між векторами	Double	double[] categorV, double[] textV, doubletextLen	Вектор tf-slfцентроїду категорії, вектор tf-slf тексту, модуль вектору тексту

3.1.4 Керівництво користувача

Перед початком роботи з ПЗ необхідно розпакувати zip-архів ClassificApp.zip в бажану директорію. Архів має наступний вміст:

– директорія workload – директорія, в якій знаходяться файли з контекстними правилами та розміченою вибіркою, яка позбавлена омонемії (опис формату файлів з вхідними даними дивитися в пункті 3.1.1);

– файл ClassificApp.jar – Java-архів, в якому міститься ПЗ;

– файл ClassificApp.bat – пакетний файл, в якому міститься команда для запуску ПЗ;

– файл config.json – JSON файл з розміченою вибіркою, яка позбавлена омонемії (приклад JSON файлу дивитися в пункті 3.1.1).

Вміст архіву ClassificApp.zip наведений на рисунку 3.4.

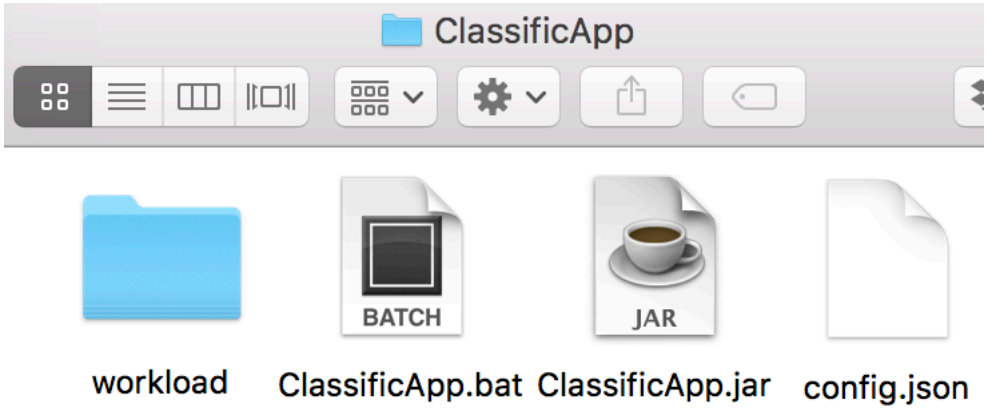

Рисунок 3.4 – Вміст архіву ClassificApp.zip

У файлі config.json необхідно відредагувати шлях до директорії, в якій знаходяться директорії з файлами, що містять показники файли з контекстними правилами та розміченою вибіркою, яка позбавлена омонімії.

Для запуску ПЗ необхідно запустити файл ClassificApp.bat (файл ClassificApp.bat та ClassificApp.jar мають бути в одній директорії), після чого відкриється головне вікно ПЗ (рисунок 3.5).

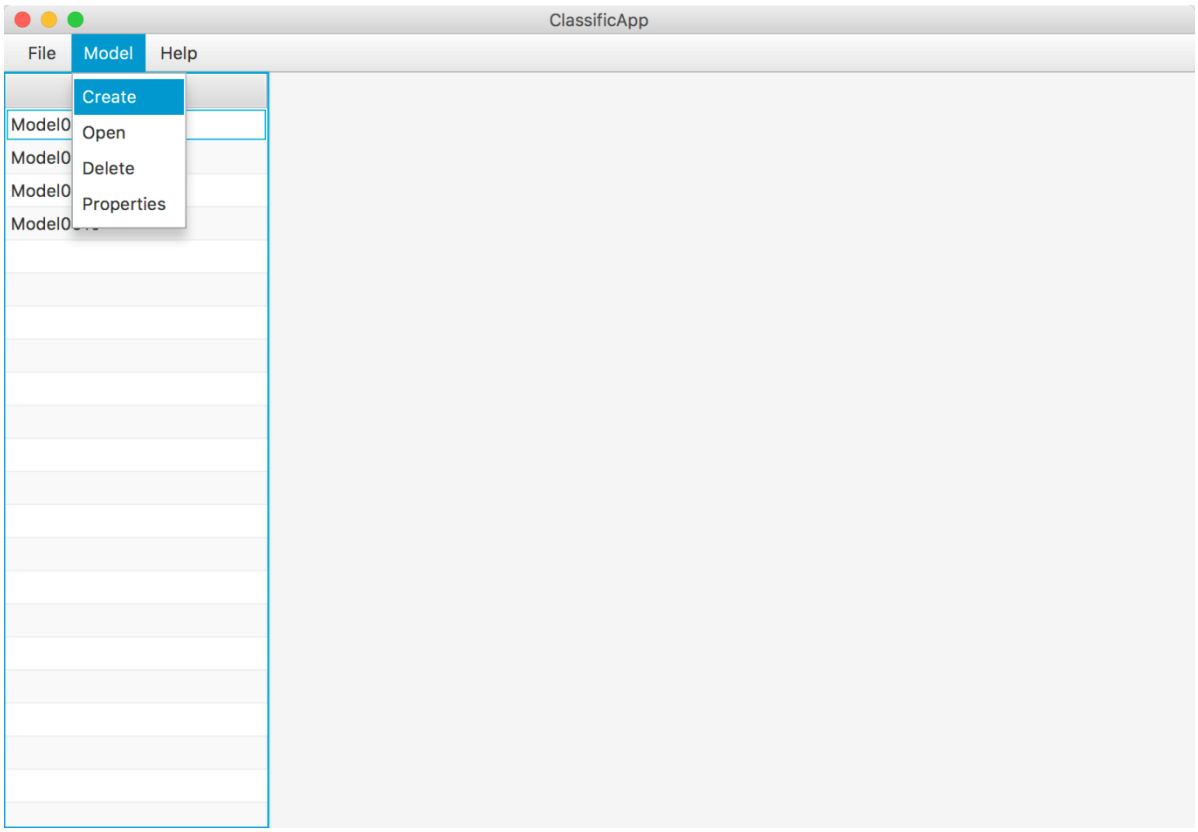

Рисунок 3.5 – Головне вікно ПЗ

Для створення класифікаційної моделі необхідно в головному меню обрати «Model>Create», після чого буде відкрите вікно для завантаження навчальної вибірки (рисунок 3.6).

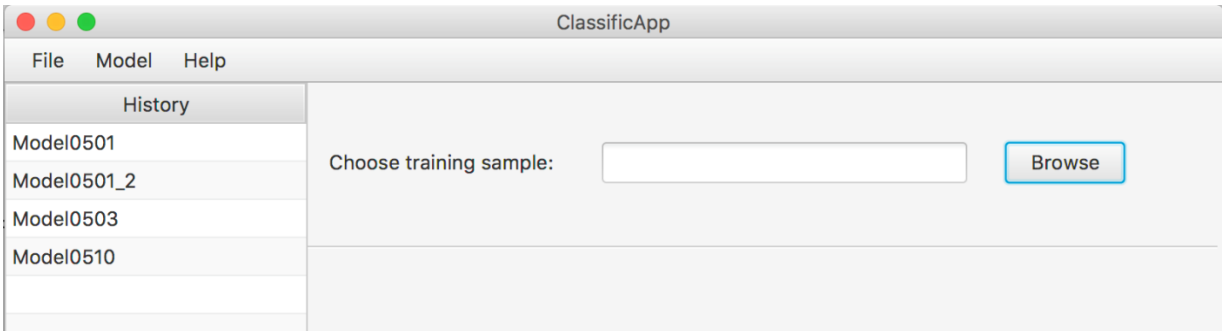

Рисунок 3.6 – Вікно завантаження навчальної вибірки

Після чого необхідно дочекатися завершення створення класифікаційної моделі, час до завершення і прогрес створення буде виведений на екран (рисунок 3.7).

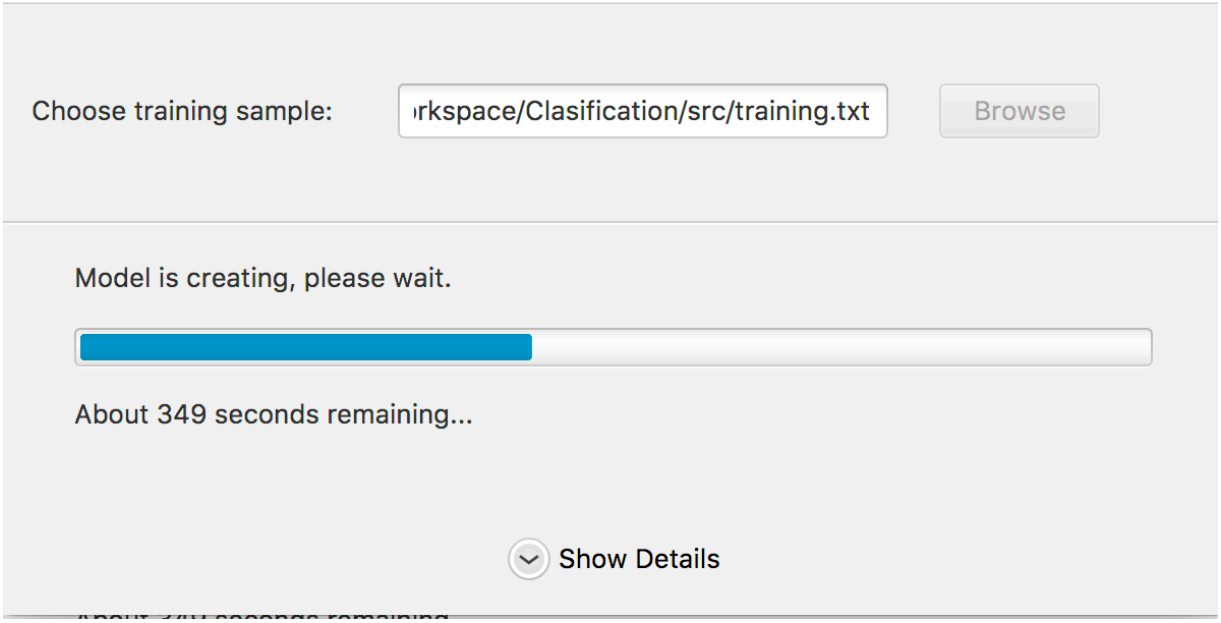

Рисунок 3.7 – Створення класифікаційної моделі

Після створення моделі, буде виведено діалогове вікно–сповіщення про успішне створення моделі (рисунок 3.8)

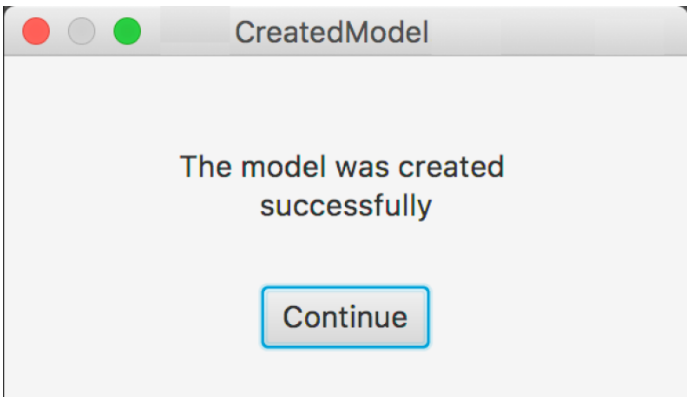

Рисунок 3.8 – Вікно– сповіщення про успішне створення класифікаційної моделі

Створена модель буде додана в список доступних моделей в лівій частині екрану. Для того щоб провести класифікацію необхідно зі списку вибрати відповідну модель, та в головній частині екранної форми обрати тестову вибірку (рисунок 3.9).

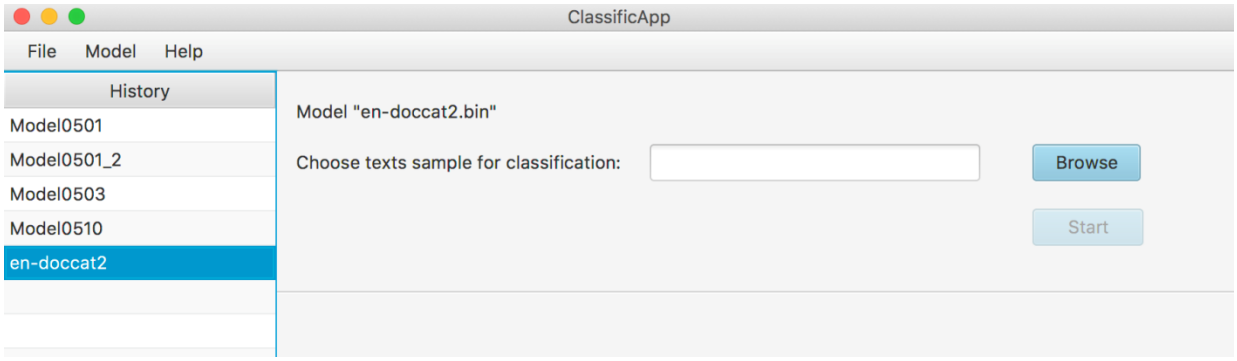

Рисунок 3.9 – Вікно завантаження тестової вибірки

Після чого буде виконана класифікація за обраною моделлю(в прикладі en-doccat2.bin). Прогрес класифікації та час, який залишився до завершення буде виведений на екранну форму (рисунок 3.10).

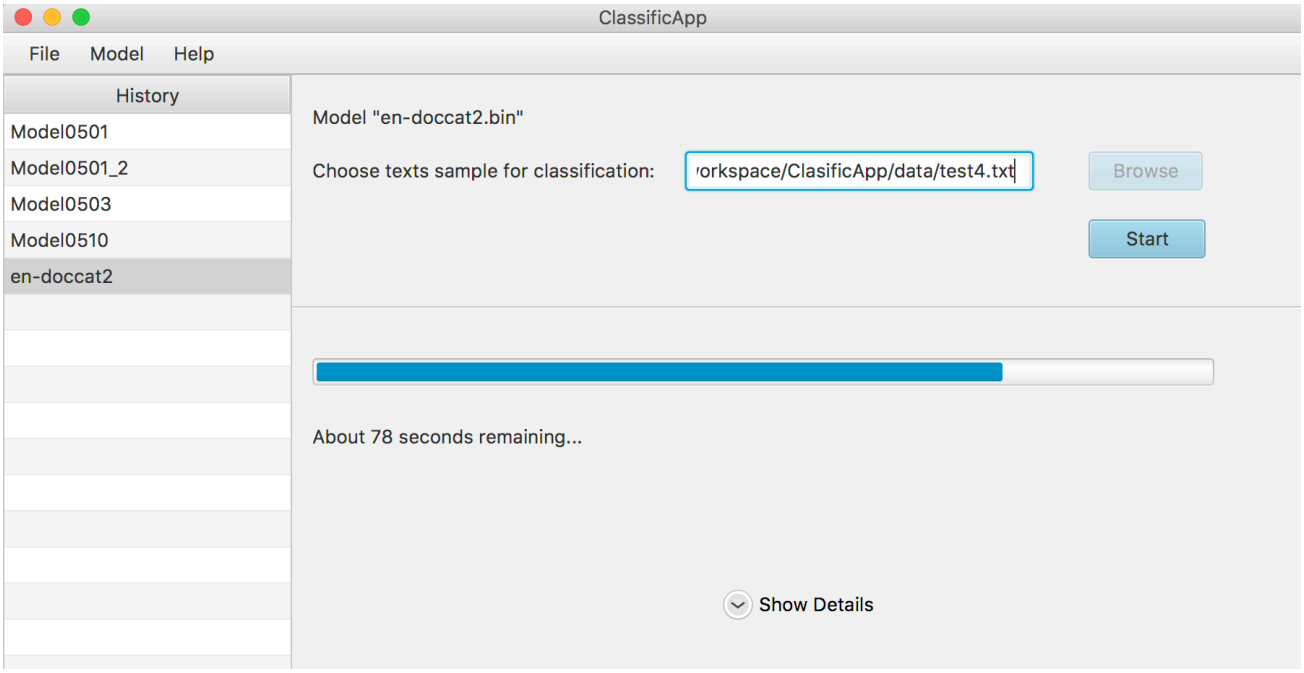

Рисунок 3.10– Процес класифікації текстових масивів інформації

По завершенню класифікації буде виведений перелік прокласифікованих текстів у вигляді таблиці з двома колонками: CategorytaText.

Отримані результати можна зберігти у форматі .csv, натиснувши на кнопку «Save», або перейти до нової класифікації, натиснувши кнопку «NewClassification» (рисунок 3.11).

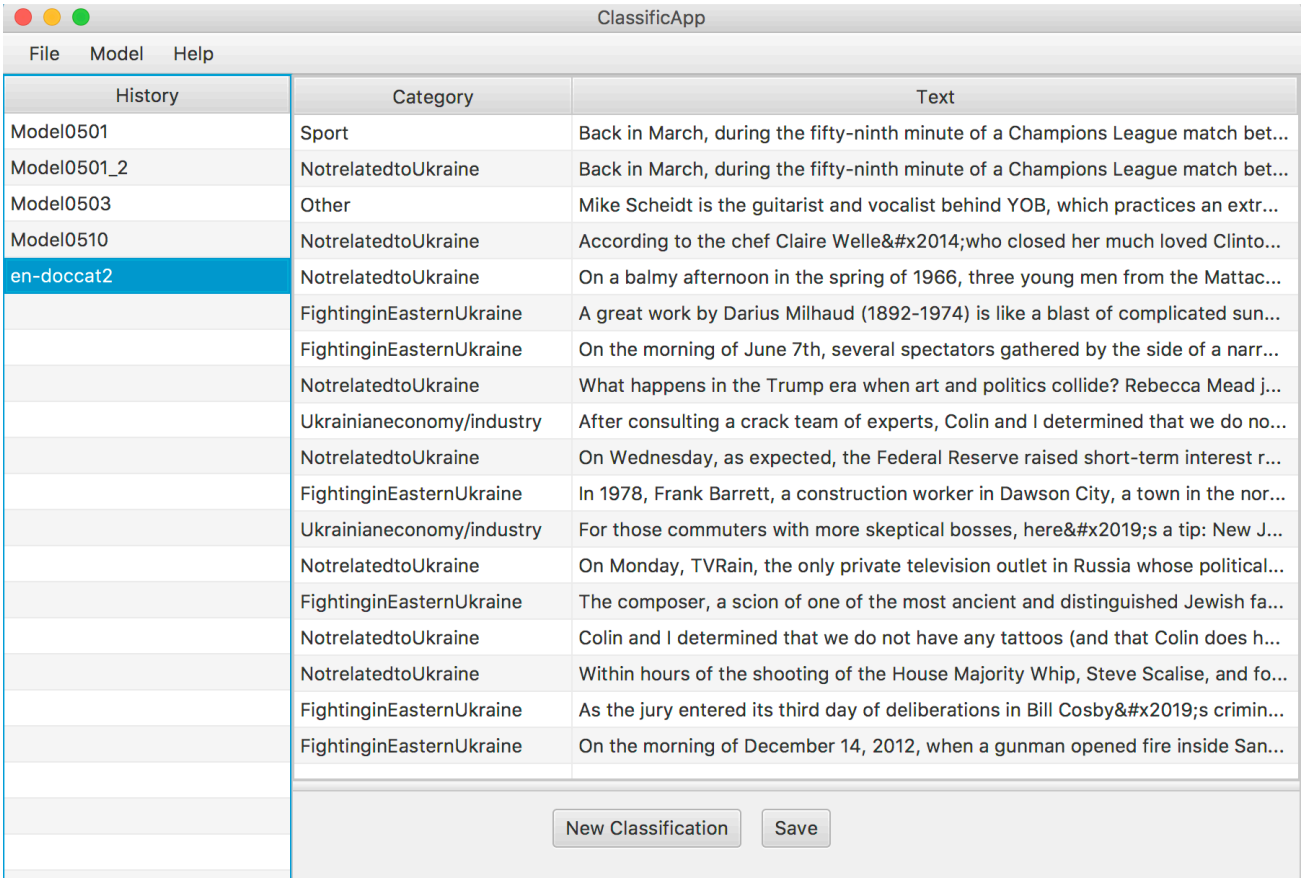

Рисунок 3.11 – Результат класифікації

Висновок до розділу

У даному розділі наведений опис форматів вхідних і вихідних даних. Наведений приклад вмісту JSON файлу з промаркерованою вибіркою, позбавленої омонімії. Описані вимоги до вхідних файлів з навчальною та тестовими вибірками, наведені приклади відповідних файлів.

Детально описані переваги та недоліки засобів розробки інформаційної технології автоматичної класифікації неструктурованих текстових масивів інформації.

Наведена схема структурна класів ПЗ та специфікація функцій ПЗ. Наведено детальне керівництво користувача.

4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Порядок проведення досліджень

Як було відзначено в розділі 1.5 основними метриками, які використовують для оцінки якості класифікації є точність (precision) та повнота (recall), або їх модифікації.

Планується провести дослідження на двох різних наборах даних. Перший, як було відзначено в розділі 3.1, взято з моніторингового проекту «ОКО»[57] та проекту «Розумне місто» [58]. Так як цей проект націлений на аналіз згадувань про Україну за кордоном, то набір текстових масивів є досить різноманітним за жанрами та тематикою, але здебільшого містить тексти з електронних журналів в публіцистичному стилі. Другий набір текстових даних взятий з фінансових звітів, планів розподілу бюджету тощо. За допомогою того, що буде використано різні види наборів текстів, можна буде оцінити поведінку модифікованого алгоритму автоматичної класифікації текстової інформації на різних даних.

З проекту «ОКО» була взята навчальна вибірка в 5000 прокласифікованих текстів, в якій зустрілися категорії, приведені в таблиці 4.1

Тестова вибірка з проекту «ОКО» складала 700 текстів, які належали класам, зазначеним в таблиці 4.1.

Для обчислення точності та повноти класифікації модифікованим алгоритмом використаємо матрицю неточності. Цей підхід дозволяє наглядно побачити роботу класифікатора. В матриці неточностей стовпці відповідають повноті, а рядки точності. Як видно на рисунку 4.1 для даної тестової вибірки точність та повнота рівні та дорівнюють 0.93. І відповідно F-міра, розрахована за формулою 1.5 буде рівна 0.93.

Таблиця 4.1– Категорії першої навчальної вибірки

Номер п/п	Назва	Кількість текстів
1	Fighting in Eastern Ukraine	715
2	Not related to Ukraine	215
3	Crimea	145
4	Global discussions about Ukraine	198
5	Other	102
6	topic of the day	160
7	Ukrainian government and society	227
9	Sport	341
10	Ukrainian economy / industry	360
11	Humanitarian crisisin Eastern Ukraine	271
12	Ukrainian international relations	140
13	MH-17 crash	261
14	Sanctions against Russia	490
15	Military / humanitarian aid for Ukraine	185
16	Peace talks	450
17	Prisoners of war	300
18	Ukrainian economy	540

	0.93	0.99	0.89	0.96	0.99	0.94	0.81	0.92	0.97	0.90	0.92	0.91	1.00	0.98	0.90	0.91	1.00	0.98	0.83
0.93		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0.96	1	70					1								1				1
0.80	2		8					1											1
0.93	3			25										1				1	
0.98	4				98		1									1			
0.98	5	1				48													
0.87	6						13			2									
0.92	7		1					12											
0.94	8					3			66						1				
1.00	9									43									
0.96	10						1				24								
0.95	11								1			21							
0.81	12										2		13		1				
0.97	13			1										57					1
0.97	14									1					28				
0.83	15				1					1						10			
0.97	16											2					64		
0.98	17								1									54	
0.94	18									1									15

Рисунок 4.1 – Матриця неточностей для модифікованого алгоритму класифікації на даних проекту «ОКО»

Для оцінки роботи алгоритму на фінансових даних була взята навчальна вибірка розміром 2300 прокласифікованих текстів. Назви категорій та кількість відповідних ним текстів наведена в таблиці 4.2.

Таблиця 4.2 – Категорії другої навчальної вибірки

Номер п/п	Назва	Кількість текстів
1	Кредитування місцевого бюджету	715
2	Річний план шкіл	215
3	План спеціального фонду бюджету	245
4	Компенсаційні виплати на пільговий проїзд	498
5	Оздоровлення та відпочинок дітей	202
6	Благоустрій міст, сіл, селищ	168
7	Податкові звіти	257

Тестова вибірка для другого експерименту складала 450 текстів, які належали класам, зазначеним в таблиці 4.2.

На рисунку 4.2 наведена матриця неточностей, яка була отримана на модифікованим алгоритмом на класифікації фінансових даних.

	0.86	0.58	0.92	0.90	0.97	0.75	0.94	0.92
0.81		1	2	3	4	5	6	7
0.54	1	7		5			1	
0.96	2		132			2	2	1
0.85	3		4	45		4		
0.95	4	4			98		1	
0.87	5	1	6			48		
0.95	6				3		63	
0.52	7		1			10		12

Рисунок 4.2 – Матриця неточностей для модифікованого алгоритму класифікації на фінансових даних

Як видно з рисунку 4.2 результуюча точність склала 0.81, а результуюча повнота склала 0.86. І відповідно F-міра, розрахована за формулою 1.5 буде рівна 0.834. З чого можна зробити висновок, що якість роботи алгоритму на фінансових даних погіршується.

4.2 Оцінка впливу алгоритму позбавлення омонімії на якість класифікації

Протестуємо на тих самих наборах даних (див. таблиці 4.1 та 4.2) роботу класифікатора, в якому модель основана на класифікаторі Роші, в якості алгоритму визначення ваг ознак документу реалізовано TF-SFL, як і в модифікованому алгоритмі класифікації, але не застосовуючи гібридний алгоритм зняття омонімії.

На рисунках 4.3 та 4.4 приведено матриці неточностей результату роботи такого алгоритму на даних з проекту «ОКО» та на фінансових даних відповідно.

	0.89	0.94	1.00	0.77	0.96	0.94	0.88	0.77	0.88	0.85	0.92	0.65	0.92	0.98	0.94	0.91	0.94	0.98	0.78
0.87		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0.96	1	67					2												1
0.64	2		9			1						3			1				
0.95	3			20		1													
0.95	4				95				4										1
0.91	5	1				48				2							2		
0.82	6						14		2										1
0.71	7	1		2				10										1	
0.94	8	1							60			3							
0.93	9				2					41						1			
0.92	10			1							24		1						
0.94	11					1						15							
0.71	12	1			2					2			12						
0.97	13							1				1		57					
0.85	14			3							2				29				
0.83	15							2								10			
0.97	16								1			1					60		
0.95	17								1						1			54	1
0.70	18									3				1			2		14

Рисунок 4.3 – Матриця неточностей для алгоритму класифікації без зняття омонімії на даних проекту «ОКО»

Відповідно F-мірааалгоритму класифікації без зняття омонімії на даних проекту «ОКО» рівна 0.879, що на 0.05 нижче отриманого показника модифікованим алгоритмом.

	0.84	0.58	0.92	0.90	0.94	0.75	0.94	0.86
0.79		1	2	3	4	5	6	7
0.47	1	7		5		2	1	
0.96	2		133		3		2	
0.83	3		5	45		4		
0.96	4	3			95		1	
0.87	5		5			48		2
0.93	6	2			3		63	
0.52	7		1			10		12

Рисунок 4.4 – Матриця неточностей для алгоритму класифікації без зняття омонімії на фінансових даних

Відповідно F-міра алгоритму класифікації без зняття омонімії на фінансових даних буде рівна 0.814, що приблизно на 0.02 гірше отриманого результату модифікованим алгоритмом.

Отже, можна зробити висновок, що застосування гібридного алгоритму, описаного в розділі 2.3, покращує показники якості класифікації текстової інформації як на текстах публіцистичного жанру так і на фінансових даних.

4.3 Порівняння розробленого алгоритму з методом MaxEnt

Порівняємо роботу модифікованого алгоритм класифікації у з методом MaxEnt. Для цього протестуємо роботу методу MaxEnt на даних наведених в таблицях 4.1 та 4.2.

На рисунку 4.5 наведено матрицю неточностей результату класифікації даних з проекту «ОКО» методом MaxEnt.

Як видно з рисунку 4.5 для MaxEnt результуюча точність рівна 0.89, результуюча повнота – 0.9. Тож F-міра методуMaxEnt на даних проекту «ОКО» буде рівна 0.895, що приблизно на 0.035 гірше отриманого результату модифікованим алгоритмом.

	0.90	0.97	0.89	0.88	0.98	0.94	0.81	0.77	0.88	0.85	0.92	0.65	0.92	0.98	0.94	1.00	1.00	0.98	0.78
0.89		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0.93	1	69	1				3												1
0.62	2		8			1						3			1				
0.96	3			23		1													
0.95	4				97				4										1
0.98	5	1				48													
0.81	6						13		2										1
0.77	7			2				10										1	
0.94	8	1							60			3							
0.95	9				2					41									
0.92	10			1							24		1						
0.94	11					1						15							
0.75	12									4			12						
0.97	13							1				1		57					
0.94	14										2				29				
0.85	15							2								11			
0.97	16								1			1					64		
0.95	17								1						1			54	1
0.78	18									3				1					14

Рисунок 4.5 – Матриця неточностей для класифікації методом MaxEnt на даних проекту «ОКО»

Як видно з рисунку 4.6 для MaxEnt результуюча точність рівна 0.81, результуюча повнота – 0.84. Тож F-міра методу MaxEnt на фінансових даних буде рівна 0.825, що приблизно на 0.01 гірше отриманого результату модифікованим алгоритмом.

	0.84	0.50	0.96	0.90	0.95	0.72	0.96	0.92
0.81		1	2	3	4	5	6	7
0.60	1	6			2	2		
0.96	2		137	4			2	
0.85	3	2		45	1	4		1
0.96	4	2		1	96		1	
0.90	5		5			46		
0.96	6		1		2		64	
0.46	7	2				12		12

Рисунок 4.6 – Матриця неточностей для класифікації методом MaxEnt на фінансових даних

Зведена таблиця результатів ефективності класифікації модифікованого алгоритму в порівнянні з методом MaxEnt та класичним класифікатором Роші наведена в таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз модифікованого алгоритму з існуючими методами класифікації

		Модифікований алгоритм	Класифікатор Роші	MaxEnt
Дані з проекту «ОКО»	Точність	0.93	0.87	0.89
	Повнота	0.93	0.89	0.90
	F-міра	0.93	0.879	0.895
Фінансові дані	Точність	0.81	0.79	0.81
	Повнота	0.86	0.84	0.84
	F-міра	0.834	0.814	0.825

Висновок до розділу

У даному розділі було наведено порядок проведення досліджень, визначені навчальні та тестові вибірки, на яких проводилось дослідження. Було виконано порівняння якості класифікації на різних даних – статей журналів та фінансових даних. Отримано, що для фінансових даних результат якості F-міри трохи нижчий.

Проведено порівняльний аналіз якості класифікації модифікованим методом з аналогічним класифікатором без застосування гібридного методу зняття омонімії та з методом MaxEnt. Отримано, що на обидвох наборах даних модифікований алгоритм класифікації дає кращий результат.

ЗАГАЛЬНІ ВИСНОВКИ

При виконанні магістерської дисертації було проаналізовано предметне середовище автоматичної класифікації неструктурованих текстових масивів інформації. Проведено алгоритми та методи машинного навчання для вирішення кожного з етапів задачі автоматичної класифікації текстової інформації, а саме: попередньої обробки та індексації документів, зважування термів та зменшення розмірності множини, побудови та навчання класифікатора та оцінки якості класифікації. Освітлено сучасні проблеми попередньої оброки текстів і для дослідження та пошуку методів позбавлення від неї була обрана морфологічна омонімія.

Була наведена формальна постановка задачі класифікації текстових масивів інформації, розглянуто основні підходи до зняття морфологічної омонімії та розвиток наукової думки стосовно рішення цієї проблеми в англомовних текстах. Розглянуто існуючі моделі представлення текстової інформації в класифікаторі, та для інформаційної технології, яка була розроблена, обрана модель представлення текстової інформації – лінгвістична розмітка.

Розроблено модифікований метод попередньої оброки текстових масивів інформації на основі статистичного алгоритму Вітербі з долученням детермінованих правил зняття омонімії на основі словника контекстів. Для того щоб зменшити час роботи алгоритму Вітербі пропонується застосовувати перевірку на наявність контекстних правил для поточного слова, і якщо відповідного правила не існує, то виконувати пошук леми та тегів статистичним алгоритмом.

Розроблено модифікований алгоритм загальної класифікації, в якому в якості алгоритму визначення ваг ознак документів обрано метод TF-SLF, який дає найкращі результати в порівнянні з TF-IDF та PMI; а в якості методу побудови та навчання класифікатора обрано класифікатор Роші.

Розроблена інформаційна технологія автоматичної класифікації неструктурованих текстових масивів інформації за допомогою мови програмування Java та фреймворку створення графічного інтерфейсу JavaFX. Виконана програмна реалізація модифікованого алгоритму автоматичної класифікації неструктурованих текстових масивів інформації. Представлена схема структурна класів програмного забезпечення та специфікацію функцій. Наведено детальне керівництво користувача.

Проаналізовано ефективність роботи інформаційної технології на двох різних наборах. Для текстів з журналів, які були взяті для тестування розробленої технології з проекту «ОКО», були отримані кращі значення F-міри в порівнянні з класифікацією фінансових даних. На цих же наборах текстових даних виконана класифікація методом MaxEnt та методом на основі класифікатора Роші, але без інтегрування модифікованого алгоритму попередньої обробки текстової інформації. Отримані результати показують, що модифікований метод дає кращі результати якості класифікації (як за точністю так і з повнотою) в порівнянні з іншими.

ПЕРЕЛІК ПОСИЛАНЬ

1. Олійник Ю., Аналіз методів визначення ваг ознак текстових документів. [Текст] / Ю.О. Олійник, Д.О. Катющенко // Журнал науковий огляд № 3(46), 2018 – С. 112–123.

2. Катющенко Д. Дослідження Методу Максимума Ентропії Для Класифікації Текстів На Прикладі Apache Opennlp Doccat [Текст]: / Д. Катющенко, Ю. Олійник // Математичне та імітаційне моделювання систем. МОДС ’2017” – 2017.

3. Катющенко Д. Дослідження методів попередньої обробки та векторизації текстових документів [Текст]: / Д. Катющенко, Ю. Олійник //Матеріали наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка» – ІОТ-2018, 24– м. Київ: С. 193-195

4. Ханько Г. В. Огляд алгоритмів текстового аналізу даних[Текст]: / Г.В. Ханько // “Інформатика та обчислювальна техніка” – ІОТ-2017– Київ: 2017. С. 17 – 20.

5. Батура Т.В. Основы обработки текстовой информации [Текст]: / Т.В.Батура, М.В. Чаринцева // Учебное пособие –Институт систем информатики им. А.П. Ершова СО РАН– Новосибирск: 2016– С.45.

6. Козлов П.Ю. Способы представления текстовой информации при автоматизированном рубрицировании коротких текстовых документов [Текст]: / П.Ю. Козлов // Программные продукты и системы / Software&SystemsT.30. №4, 2017 – С. 678–683.

7. Файловый архив студентов StudFiles [Електронний ресурс] // Режим доступу: https://studfiles.net/preview/4533558/page:12/

8. Joachims T. “Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms” [Текст]: / T. Joachims // Kluwer Academic Publishers Norwell, MA, USA: 2002.

9. Батура Т.В. Методы автоматической классификации текстов [Текст]: / Т.В. Батурамеждународный научно-практический журнал программные продукты и системы №1 : 2017. С. 85 – 99.

10. Воронцов К.В. Лекции по линейным алгоритмам классификации [Електронний ресурс] // Режим доступу: http://www.machinelearning.ru/wiki/images/6/68/voron-ML-Lin.pdf

11. Оценка классификатора (точность, полнота, F-мера) [Електронний ресурс] / Суровая реальность – 2012. – Режим доступу: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html

12. Ballard D.H. Computer Vision [Текст]: / D.H. Ballard, C.M. Brown C.M. // Prentice Hall Inc., 1982. – 539p.

13. Mikolov T. Distributed Representations of Words and Phrases and their Compositionality [Текст]: / T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean // Proceeding NIPS 13 Proceedings of the 26th International Conference on Neural Information Processing Systems – Volume 2. – 2013. – pp. 3111-3119.

14. Mikolov T. Linguistic Regularities in Continuous Space Word Representations [Текст]: / T. Mikolov, W. Yih, G. Zweig // Proceedings of NAACL-HLT 2013 – Atlanta, Georgia – 2013. pp.746–751.

15. Zhang W. A comparative study of TF*IDF, LSI and multi-words for text classification [Текст]: / W. Zhang, T. Yoshida, X. Tang // Expert Systems with Applications - Volume 38 Issue 3 - 2011, pp. 2758-2765.

16. Нугуманова А.Б Обогащение модели BAG-OF-WORDS семантическими связями для повышения качества классификации текстов предметной области [Текст] : / А.Б. Нугуманова, И.А. Бессмертный, П. Пецина, Е.М. Байбурин // “Програмные продукты и системы”. 2016. №2 (114). С. 89 – 99.

17. Sahlgren M. Usong bag-of-concept to improve the performance of support vector machines in text categorization [Текст]: / M. Sahlgren, R. Coster // Proc. of the 20th Int. Conf. on Computational Linguistics, Association for Computational Linguistics. 2004. p. 487.

18. Huang A. Clustering documents using a Wikipedia-based concept representation [Текст]: / A. Huang // Advances in Knowledge Discovery and Data Mining. SpringerBerlin: Heidelbergpubl. 2009. pp 628-636.

19. Shehata S. An efficient concept-based mining model for enhancimg text clustering. [Текст]: / S. Shehata, F. Karray, M. Kamel // IEEE Transactions on Knowledge and Data Engineering. 2010, vol. 22, #10. pp.1360-1371.

20. Egozi O. Concept-based indexing text information retrieval [Текст]: / O. Egozi, S. Markovitch, E. Gabrilovich // ACM Transactions on Information System (TOIS). 2011, vol.29 #2. p 8.

21. Boubekeur F. Concept-based indexing in text information retrieval [Текст]: / F. Boubekeur, W. Azzoug // Int. J. Comput. Sci. Inf. Technol. 2013, vol. 5 pp. 119-136.

22. Huang L. Learning a concept-based document similarity measure. [Текст]: / L. Huang // Journ. of the American Society for Information Science and Technology. 2012, vol. 63, #8. pp.1593-1608.

23. Wang P. Building semantic kernels for text classification using Wikipedia [Текст]: / P. Wang, C. Domeniconi // Proc. of the 14th ACM SIGKDD Intern. Conf. on Knowledge discovery and data mining: ACM Publ., 2008, pp.713-721.

24. Turdakov D.Y. Word sense disambiguation methods. [Текст]: / D.Y. Turdakov // Programming and Computer Software. 2010, vol. 36, #6, pp.309-326.

25. Dinh D. Combining global and local semantic contexts for improving biomedical information retrieval. [Текст]: / D. Dihn, L. Tamine // Advances in Information Retrieval – Berlin: Heidelberg Publ. – 2011. pp. 375-386.

26. Carrillo M. Concept Based Representations as complement of Bag Of Words in Information Retrieval [Текст]: / M. Carrillo, A. Lopez-Lopez // Artificial Intelligence Applications and Innovations – Berlin: Heidelberg Publ. – 2010. pp. 154-161.

27. Deerwester S.C. Indexing by latent semantic analysis [Текст]: / S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman // JAsls - 1990, no.1, pp. 91 - 102.

28. Ju R. An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis. [Текст]: / R. Ju // — 2015 IEEE Int. Conf. on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Auto-nomic and Secure Computing; Pervasive Intelligence and Computing. Liverpool, UK, 2015, pp. 2276–2283.

29. Agirre E. A study on similarity and relatedness using distributional and WordNet-based approaches. [Текст]: / E. Agirre // Proc. of Human Language Technologies: the 2009 Annual Conf. of the North American Chapter of the Association for Computational Linguistics. – ACL Publ. – 2009. pp. 19 – 27.

30. Mashechkin I. Automatic text summarization using latent semantic analysis. [Текст]: / I. Mashechkin, M. Petrovsky, D. Popov, D. Tsarev // Programming and Computer Software. – 2011, vol. 37, no. 6, pp. 299-305.

31. Mashechkin I. Methods of text fragment relevance estimation based on the topic model analysis in the text summarization problem. [Текст]: / I. Mashechkin, M. Petrovsky, D. Tsarev // Vychislitelnye metody i programmirovanie [Computing Methods and programming]. - 2013, vol. 14, no. 1, pp. 91-102.

32. Xiang Zhang Character-level convolutional networks for text classification. [Текст]: / Junbo Zhao, Yann LeCun. // Proc. Neural Inform. Processing Systems Conf. (NIPS 2015). Montreal, Canada - 2015.

33. Victor S. Miller Variations on a theme by Ziv and Lempel, Combinatorial algorithms on words. [Текст]: / Victor S. Miller, Mark N. Wegman – Springer Berlin Heidelberg, 1985. – pp. 131-140.

34. Гасфилд Д. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология [Текст]: / Д. Гасфилд Пер. с англ. И. В. Романовского. — 2-е изд. — СПб.: Невский Диалект, 2003. — 654 с.

35. Окулов С. М. Алгоритмы обработки строк. — М.: Бином, 2013. — 255с.

36. Ricardo Baeza-Yates Modern information retrieval [Текст]: / Ricardo Baeza-Yates, Berthier Ribeiro-Neto. — Reading, Massachusetts: Addison-Wesley Longman, 1999. — 192 с.

37. Попков М.И. Автоматическая система классификации текстов для базы знаний предприятия: магистерская дисертация — М: 2014. - 57 с.

38. Victor S. Miller Variations on a theme by Ziv and Lempel, Combinatorial algorithms on words. [Текст]: / Victor S. Miller, Mark N. Wegman – Springer Berlin Heidelberg, 1985. – pp. 131-140.

39. Corder G. W. Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. [Текст]: / G. W. Corder, D. I. Foreman// — New York: Wiley, 2009.

40. Yiming Yang A comparative study of of feature selection in text categorization [Текст]: / Yiming Yang Jan O. Pedersen // ICML ‘97 Proceedings of the fourteenth international conference on machine learning - San Francisco, CA, USA, 1997 — pp. 412-420.

41. Zhang C. Automatic Keyword Extraction from Documents Using Conditional Random Fields [Текст]: / C. Zhang, H. Wang, Y. Liu, D. Wu, Y. Liao, B. Wang // Journal of Computational Information Systems 4:3. 2008. pp. 1169-1180.

42. Sharma A. R. Literature survey of statistical, deep and reinforcement learning in natural language processing Sign In or Purchase [Текст]: / A. R. Sharma, P. Kaushik // 2017 International Conference on Computing, Communication and Automation *(ICCCA)*, Greater Noida, 2017, pp. 350-354.

43. Ghahramani Z. An introduction to hidden Markov models and Bayesian networks. [Текст]: / Z. Ghahramani // International Journal of Pattern Recognition and Artificial Intelligence, 15(01). 2001. pp. 9–42.

44. Manning R.R. Introduction. [Текст]: R.R. Manning (eds) Retrieving the Radical Tillich. Radical Theologies. Palgrave Macmillan— NewYork— 2015. pp. 1-19.

45. Suthaharan S. Support Vector Machine. In: Machine Learning Models and Algorithms for Big Data Classification. [Текст]: / S. Suthaharan // Integrated Series in Information Systems, vol 36. Springer — Boston, MA — 2016. pp. 207-235.

46. Гришанов К.М. Метод классификации k - nn и его применение в распознавании символов [Текст]: / К.М. Гришанов, Ю.С. Белов // Фундаментальные проблемы науки. — Тюмень: НИЦ АЭТЕРНА — 2016. 30-33 с.

47. Manning D. Christopher Introduction to information retrieval [Текст]: / Christopher D. Manning, Prabhakar Raghavan // Cambridge University Press - 2008.

48. Maimon Oded Z. Data Mining With Decision Trees: Theory And Applications (2nd Edition) [Текст]: / Maimon Oded Z, RokachLior — 2014. 304 p.

49. Смирнова О.С. Выбор топологии нейронных сетей и их применение для классификации коротких текстов [Текст]:/ О.С. Смирнова, В.В. Шишков. International Journal of Open Information Technologies, vol. 4.//– 2016, pp. 50–54.

50. Гавриленко О. В. Огляд та аналіз алгоритмів TEXT MINING / О.В. Гавриленко, Ю.О. Олійник, Г. В. Ханько. // Управління проектами, системний аналіз і логістика. – К.: НТУ, 2017. – Вип.

51. Apache™ OpenNLP [Електронний ресурс] // Режим доступу: http://opennlp.apache.org/

52. uClassify [Електронний ресурс] // Режим доступу: https://www.uclassify.com/

53. Ленкин А.В. Исследование систем для TextMining [Текст]: / А.В. Ленкин, Р.И. Баженов // ISSN 2414-4487. Постулат. №1. – 2017 –с. 326-340

54. Joachims T. “Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms” [Текст]: / T. Joachims // Kluwer Academic Publishers Norwell, MA, USA. –2002.

55. Рысаков С.В. Статистические методы снятия омонимии [Текст]: / С.В. Рысаков, С.Э. Клышинский // Новые информационные технологии в автоматизированных системах. – 2015. – С. 555 – 563.

56. Зеленков Ю.Г. Вероятностная модель снятия морфологической омонимии на основе нормализующих подстановок и позиций соседних слов [Елекстронний ресурс] // режим доступу: http://www.dialog-21.ru/media/2444/zelenkov_segalovich.pdf

57. Моніторинговий проект “ОКО” [Електронний ресурс] / Режим доступу: http://www.ukroko.org

58. Електронне врядування «Розумне місто» [Електронний ресурс] / Режим доступу: https://rozumnemisto.org

59. Мова програмування Java [Електронний ресурс] / Режим доступу: https://java.com/ru/

60. Платформа для створення графічного інтерфейсу додатків JavaFX [Електронний ресурс] / Режим доступу: http://www.oracle.com/technetwork/java/javafx/

61. Model View Controller (MVC) [Електронний ресурс]/ Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller

62. Інтегроване середовище розробки Eclipse [Електронний ресурс] / Режим доступу: http://www.eclipse.org/

63. JavaFX SceneBuilder [Електронний ресурс] / Режим доступу:https://o7planning.org/en/10621/install-javafx-scene-builder-into-eclipse

ДОДАТОК А Графічний матеріал

ПЛАКАТ 1 Схема класифікації

ПЛАКАТ 2 Блок-схема алгоритму індексації текстових документів

ПЛАКАТ 3 Блок-схема загального алгоритму класифікації

ПЛАКАТ 4 Схема структурна класів програмного забезпечення

ПЛАКАТ 5 Схема структурна послідовності процесу індексації текстових документів

ПЛАКАТ 6 Копії екранних форм

ПЛАКАТ 7 Результати досліджень